Insinöörityökalun käyttöliittymän suunnittelun haasteet

Mikko Manninen

Tampereen yliopisto
Informaatiotieteiden yksikkö /
Vuorovaikutteinen teknologia
Pro gradu -tutkielma
Ohjaaja: Markku Turunen
11.6.2015

Tampereen yliopisto
Informaatiotieteiden yksikkö / Vuorovaikutteinen teknologia
Mikko Manninen: Insinöörityökalun käyttöliittymän suunnittelun haasteet
Pro gradu -tutkielma, 76 sivua
Kesäkuu 2015

Tiivistelmä

Kiireisessä tuotekehitystyössä on harvoin keskeisimpänä tavoitteena luoda loppuun asti hiottua käyttöliittymää, kun luodaan testausohjelmistoa tiettyyn tarpeeseen. Tästä syystä testausohjelmistojen käyttöliittymät muistuttavat enemmänkin väliaikaiskäyttöliittymää kuin loppukäyttäjälle suunnattua hiottua käyttöliittymää.

Tutkielmassa tutustutaan käytettävyyden perusteisiin ja sen vaikutuksesta ohjelmiston ja käyttäjän vuorovaikutuksen tehokkuuteen. Tämän seikan huomioimiseksi työkalun suunnittelutyössä tutkielmassa esitellään GOMS-malli, jonka avulla käytettävyys ja sen vaikutus tehokkuuteen voidaan huomioida suunnittelun aikana ennen varsinaista toteutusta.

Insinöörityökalun käyttöliittymäesimerkkeinä tutkielmassa käytetään OptoFidelity Oy:n kehittämiä *WatchDog* ja *Scroll Performance Analyser(SPA)* mittaustyökaluja, joiden yhteydessä GOMS-mallin käyttöä konkreettisen kohteen kanssa esitellään. Mallin toimintaa esitellään tekemällä arvio ohjelmistoon ehdotettujen parannusehdotusten vaikutuksesta. Lähtötilanteeseen nähden mallin avulla voitiin laskea 16 prosentin ajallinen säästö esimerkkitehtävän suorituksessa.

Tutkielman lopussa käydään läpi Nielsenin heuristiikkojen tarjoamaa ajatusmaailmaa, jonka avulla pahimmat käytettävyysongelmat insinöörityökaluissa voitaisiin välttää tai korjata.

Sisällys

1 Johdanto

Käyttöliittymien lisääntyminen yleisen teknologiakehityksen mukana on nostanut käyttöliittymän suunnittelun ja sen synnyttämän käyttökokemuksen (UX) tutkijoiden huomioon. Huomio on ansaittua käyttöliittymille ja niiden synnyttämille käyttökokemuksille, koska ne ovat osa arkea nykypäivänä. Työelämässä ei ole enää monta paikkaa, jossa ihminen ei joutuisi tekemisiin erinäköisten käyttöliittymien kanssa.

Käyttöliittymä ei ole pelkästään rajattu tietokonemaailmaan, vaan käyttöliittymän voi löytää esimerkiksi niinkin yksinkertaisesta tuotteesta kuin vasaran varressa olevasta kahvasta. Kyseinen kahva on vasaran käyttöliittymä, jonka avulla käyttäjä voi suorittaa vasaralla sille tyypillisiä tehtäviä. Fyysisesti isompana esimerkkinä voitaisiin pitää esimerkiksi supermarkettia, jonka käyttöliittymä muodostuu liiketilasta myymäläkalusteineen ja siellä työskentelevästä henkilökunnasta. Käyttäjä, joka on tässä tapauksessa asiakas, käyttää kaupan tarjoamia palveluita kaupan tarjoaman käyttöliittymän avulla.

Hyvän käyttöliittymän kriteerit riippuvat käyttäjästä. Supermarket-esimerkin tilanteessa hyvän käyttöliittymän kriteerit vaihtelevat suuresti esimerkiksi rullatuolilla ja normaalisti liikkuvan henkilön välillä. Normaalisti liikkuvalle henkilölle esimerkiksi maitotuotehyllyä voi olla helppo käyttää fyysisesti, mutta pyörätuolilla liikkuvalla voi puolestaan olla suuriakin ongelmia ylettyäkseen ylemmille hyllyille. Pyörätuolilla liikkuvalle henkilölle sopiva hyllykorkeus puolestaan voisi olla hankala normaalilla liikuntakyvyllä olevalle henkilölle. Normaalisti supermarketit on hyllyjensä ja liiketilojensa puolesta suunniteltu valtaenemmistölle. Sama enemmistön mukaan suunnittelu näkyy myös yksinkertaisessa tuotteessa kuten esimerkiksi saksissa. Yleisesti kaupoissa myytävät sakset on suunniteltu oikeakätisille, joten ne ovat hankalia käytettäviä vasenkätisille. Edellä mainittujen esimerkkien avulla voi hahmottaa mieleensä kuvan siitä, että käyttöliittymä vaikuttaa kohteen käytettävyyteen, jonka käyttäjä kokee.

Tietotekniikan parissa käytettävyyden tulee myös ottaa huomioon ihmisten erilaiset fyysiset rajoitteet. Esimerkkinä voidaan mainita puna-vihersokeudesta kärsivät henkilöt, joille voi tuottaa vaikeuksia tulkita käyttöliittymässä olevia punaisella ja vihreällä värillä tilaansa ilmaisevia tilalamppuja. Tällaisessa tapauksessa oikein tulkitsemista voidaan tukea jollain symbolilla, kuten esimerkiksi rastilla

punaisen värin palaessa. Käyttäjien erilaiset tiedolliset ja kulttuurisidonnaiset taustat asettavat myös haasteensa käyttöliittymän suunnittelussa.

Insinöörityökaluiksi miellettävissä ohjelmistoissa ja laitteissa käyttöliittymä voi poiketa hyvinkin radikaalisti kuluttajatuotteiden käyttöliittymistä. Syitä tähän hyvinkin erilaiseen lähestymistapaan käyttöliittymän osalta on monia ja niistä yhtenä esimerkin vuoksi voidaan mainita käyttäjien hyvinkin erilainen tietämys tuotteen toimintaympäristöstä. Kaikkia insinöörejä tai suunnittelijoita ei ole veistetty samasta puusta ja sen vuoksi toisen kollegan suunnittelema käyttöliittymä voi olla toiselle hyvinkin hämmentävä. Tälläisissä tilanteissa käytettävyys vaikuttaa ohjelmiston tai työkalun avulla suoritettujen tehtävien suoritustehokkuuteen. Panostamalla hiukan työkalujen ja ohjelmistojen käytettävyyteen yritys voi säästää säästyneen ajan myötä. Tämän tutkimuksen tavoitteena on tarjota tiiviissä paketissa ohjeet pahimpien käytettävyysongelmien välttämiseen insinöörityökaluiksi mielletyissä ohjelmistoissa.

1.1 Tutkimuskysymykset

Tämän tutkimuksen päämotivaattorina on kirjoittajan oma historia insinöörityökalujen parissa. Tämän historian saatossa esiintyneisiin käytettävyysongelmiin pyritään tutkielmassa löytämään ratkaisu seuraavilla tutkimuskysymyksillä:

1. Miten käytettävyys huomioidaan järjestelmässä ja sen suunnittelussa siten, että se myös tukee tehokasta työskentelyä?

2. Miten jonkin olemassa olevan työkalun käytettävyyttä voitaisiin parantaa ja sen kautta tehostaa työskentelyä?

1.2 Tutkielman sisältö

Tässä tutkielmassa käydään alkuosassa läpi käytettävyyden perusteita ja niiden vaikutusta järjestelmän ja sen käyttäjän väliseen tehokuuteen. Tämän jälkeen esitellään GOMS-malli, joka valikoitui kirjallisuutta läpikäydessäni. GOMS-mallia esittelemällä tutkielman tarkoituksena on luoda lukijalle käsitys suunnittelumetodeista, joilla käytettävyyden ja järjestelmän tehokkuus voidaan ottaa huomioon. Tutkielmassa esitellään kaksi OptoFidelity Oy:n kehittämää työkalua, joita voidaan pitää esimerkkeinä insinöörityökaluista. Työskentelyä haittaavia käytettävyysongelmia näistä kahdesta edellämainitusta työkalusta haettiin haastattelemalla henkilöi-

tä, jotka olivat käyttäneet työkaluja. Haastateltavia henkilöitä oli viisi kappaletta, ja haastattelut toteutti OptoFidelity Oy. Tutkielman loppuosassa pohditaan, miten käytettävyysongelmia ja niiden aiheuttamaa tehokkuuden laskua voitaisiin ehkäistä insinöörityökalujen kehitystyössä.

2 Käsitteiden esittely

Tässä kappaleessa käydään läpi tämän tutkielman kannalta tärkeitä käsitteitä.

2.1 Käyttöliittymä

Käyttöliittymä on käyttäjälle luotu tapa kommunikoida järjestelmän kanssa. Käyttöliittymän tulkitseminen riippuu käyttäjän käsitemalleista [Norman, 1988]. Käyttäjällä on taipumus tulkita käyttöliittymän toimintaa päässänsä ja simuloida sen toimintaa mielessään.

2.2 Vuorovaikutus

Tietotekniikan kanssa käyttäjän vuorovaikutus järjestelmän kanssa perustuu käyttäjän antamiin syötteisiin ja järjestelmän palauttamiin tulosteisiin [Norman, 1988].

2.3 Käytettävyyden käsite

Tässä tutkielmassa käytettävyyden käsite määritellään ISO 9421-11-standardin [ISO 9421, 1998] ja Nielsenin [1993] teesien avulla. ISO 9421-11-standardin mukaan käytettävyyden käsite koostuu seuraavista osatekijöistä:

- Tuloksellisuus
- Tehokkuus
- Tyytyväisyys
- Opittavuus

2.3.1 Tuloksellisuus

Tuloksellisuus on käytettävyyden osatekijä, joka puolestaan koostuu seuraavista asioista:

- Suoritettujen tehtävien prosentuaalinen määrä.
- Virheiden määrä.
- Tavoiteajassa suoritettujen tehtävien prosentuaalinen määrä.

Tuloksellisuuden tehtävänä käytettävyyden arvioinnissa on toimia yleisenä indikaattorina siitä, kuinka hyvin käyttäjä pystyy käytetyn systeemin avulla suorittamaan hänelle asetettuja tehtäviä.

2.3.2 Tehokkuus

Tehokkuus on relaatio kahden seuraavan kokonaisuuden välillä [Frokjaer et al., 2000]:

1. Tehtävien suorittamisen tarkkuus ja täydellisyys.
2. Käytettyjen resurssien määrä tavoitteeseen pääsyyn.

Nämä indikaattorit sisältävät tehtävään käytetyn ajan ja oppimisprosessin ajan.

2.3.3 Tyytyväisyys

Tyytyväisyydellä tarkastellaan käyttäjän suhtautumista tehtävän suorittamiseen käytettyyn systeemiin. Tämä näkyy käyttäjän epämukavuuden puutteena tehtävää suorittaessa ja yleisenä positiivisena suhtautumisena käytettyä systeemiä kohtaan.

2.3.4 Opittavuus

ISO 9421-11 Standardin [ISO 9421, 1998] mukaan opittavuudella tarkastellaan käytön oppimiseen kuluvaa aikaa.

2.3.5 Oppimiskäyrä

Oppimiskäyrä on ISO-standardin käytettävyyden määritelmän opittavuus osatekijän havainnollisempi esitysmuoto, jolla pyritään havainnollistamaan käyttöliittymän hallitsemiseen liittyvän oppimisprosessin vaativuutta. Oppimiskäyrän akselit muodostuvat ajasta (x-akseli) ja käyttäjän käyttöliittymän suoritustason indeksistä (y-akseli) [Nielsen, 1993]. Nielsenin määritelmien mukaan helposti opittavissa olevassa aloittelijoille suunnatussa käyttöliittymässä oppimiskäyrä nousee jyrkästi aika-akselin alkupäässä, mutta ei yleensä saavuta maksimitasoa. Tehokäyttäjille suunnatun käyttöliittymän oppimiskäyrä on puolestaan päinvastainen ja se nousee aluksi hitaasti ajan funktiona kohti maksimia.

2.4 Käyttäjäkeskeinen tuotekehitys

Käyttäjäkeskeisellä tuotekehityksellä tarkoitetaan järjestelmän suunnitteluprosessia ISO-standardin [ISO 13407, 1999] mukaisesti. Tämä tarkoittaa sitä, että järjestelmän suunnitteluprosessin keskeisenä päämääränä on tuottaa käyttäjien työtä tehostavia ja työn tuloksellisuutta lisääviä järjestelmiä.

2.5 Käyttökokemus

Sharpin ja kollegoiden [Sharp et al., 2007] mukaan käyttökokemus sisältää käyttäjän positiivisia ja/tai negatiivisia kokemuksia järjestelmästä. Käyttökokemus eroaa käytettävyyden käsitteestä siinä, että käyttökokemuksessa tutkitaan sitä, millaisena käyttäjät kokevat järjestelmän. Tämä näkökulma aiheuttaa sen, että käyttökokemusta määritteleviä ominaisuuksia on enemmän kuin käytettävyydessä. Käyttökokemukseen saattavat vaikuttaa ennen järjestelmään tutustumista muun muassa markkinointi tai yhteisöjen luomat mielikuvat järjestelmästä [Raita & Oulasvirta, 2010].

3 Käytettävyys järjestelmässä

Tutkittaessa käytettävyyden vaikutusta järjestelmällä suoritettujen tehtävien nopeuteen on hyvä ymmärtää käytettävyyden merkitys kokonaisuuden kannalta. Tässä kappaleessa avataan tätä näkökulmaa.

3.1 Käytettävyyden rooli

Käytettävyyden määritelmän avulla voidaan olettaa, että järjestelmän hyvä käytettävyys edesauttaa käyttäjän ja järjestelmän vuorovaikutusta. Tästä voidaan olettaa, että hyvällä käytettävyydellä on positiivinen vaikutus käyttäjän järjestelmällä suorittamiin tehtäviin. Tästä puolestaan seuraa looginen jatko-oletus, että käytettävyydellä on roolinsa järjestelmällä suoritettuihin tehtäviin.

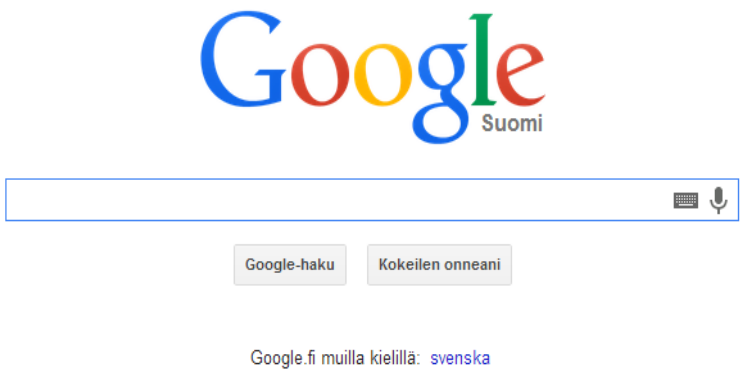
Käytettävyyden roolin vaikutus järjestelmällä suoritettuihin tehtäviin vaihtelee vuorovaikutuksen ja järjestelmän automaattisten toimintojen suorituksen suhteen perusteella. Karkeana esimerkkinä minimalistisesta roolista käytettävyydelle voisi olla esimerkiksi järjestelmä, joka huolehtii tietokoneen kiintolevyn tietojen perusteellisesta hävittämisestä. Käyttäjän ja järjestelmän vuorovaikutus pelkistyy siihen, että käyttäjä valitsee tyhjennettävän kiintolevyn ja käynnistää tyhjennysoperaation. Käynnistämisen jälkeen järjestelmä jää suorittamaan levyn tyhjennystä ja kyseinen operaatio ei vaadi käyttäjältä mitään syötettä operaation etenemiseen. Nykyaikaisen kiintolevyn tietojen perusteellinen pyyhintä vaatii useamman päällekirjoituskerran, joten pyyhintäoperaatio vaatii järjestelmältä huomattavasti aikaa. Käyttäjän ja järjestelmän välinen vuorovaikutus on tällaisessa järjestelmässä vähäistä koko tehtävän suorittamiseen kuluvaan aikaan nähden, joten käytettävyyden rooli tällaisessa tapauksessa on hyvin pieni. Vastakkaisena esimerkkinä voidaan pitää esimerkiksi laskun maksamista verkkopankissa. Tällöin käyttäjän ja järjestelmän välinen vuorovaikutus on suurimmaksi osaksi aktiivista koko maksutapahtuman ajan. Järjestelmän suorittamiin tallennustoimenpiteisiin kuluva aika on hyvin pieni osa maksutapahtuman kokonaiskestossa. Tässä esimerkissä käytettävyyden rooli on suuri tehtävän, eli maksutapahtuman, kokonaiskestossa.

Suunnitteluvaiheessa käytettävyyden rooli ja sen suuruus olisi hyvä olla tiedossa, jotta siihen osattaisiin varata oikea määrä resursseja. Järjestelmällä suoritettavien tehtävien kokonaiskeston lyhentämisen, eli suorituksen optimoinnin, kannalta eri osien rooli ja niiden suuruudet olisi syytä tietää. Näin voidaan ohjata resurssit siihen kohtaan järjestelmässä, josta saadaan eniten hyötyä tavoitteeseen pyrittäessä.

Käytettävyyden rooli voi olla myös markkinointitekninen, jolloin käytettävyyttä pidetään tuotteen myyntiargumenttina. Tällaisena esimerkkinä voidaan pitää Applen iPhonea tai Googlen hakukonetta, joiden markkinoinnissa käytettävyydellä on suuri rooli. Toinen ääripää käytettävyyden roolista markkinoinnin kannalta on tiukasti ammattikäyttöön suunnitellut järjestelmät, joiden myyntiargumentit painottuvat muualle kuin tavallisille kuluttajille suunniteltuun käytettävyyteen. Esimerkkinä voidaan mainita esimerkiksi Rohde & Schwarzin CMU200 universaali kommunikaatiotesteri (kuva 1).

Kuva 1: Rohde & Schwarz CMU200.

CMU200-testeri on suunniteltu telekommunikaation parissa työskentelevien asiantuntijoiden työkaluksi. Tästä seuraa se, että laitteen käyttöön tai pelkkään käyttötarkoituksen ymmärtämiseen jo vaaditaan suunnaton määrä pohjatietoa. CMU200-testerin käytettävyys ei muodostu normaalille kuluttajaelektroniikalle suunnatuista asioista, kuten helposta lähestyttävyydestä tai helposta muistettavuudesta. Testerin käytettävyys on suunniteltu sellaiseksi, että telekommunikaation asiantuntija pystyy suorittamaan erilaisia testejä ilman rajoittavia esteitä. Tämän vuoksi laitteen yhtenä käyttöliittymänä toimiva etupaneeli ei edusta Googlen hakukoneen aloitussivun (kuva 2) tapaista hyvin yksinkertaista käyttöliittymää.



Kuva 2: Googlen suomenkielinen hakusivu.

3.2 Käytettävyyden toteuttaminen

Käytettävyyden toteuttamisen avuksi on laadittu ISO 13407-standardi [ISO 13407, 1999], jota seuraamalla voidaan noudattaa käytettävyyteen keskittyvää prosessia. Prosessin seuraaminen auttaa tuottamaan käyttäjien työtä tehostavia, työn laatua ja tuloksellisuutta lisääviä järjestelmiä. Käytännössä prosessin noudattaminen tarkoittaa sitä, että prosessin jokaisessa vaiheessa tulee ottaa huomioon käyttäjien tarpeet, taidot ja rajoitteet. Prosessin seuraaminen on seuraavien askelien seuraamista:

1. Käyttötilanteen ymmärtäminen ja toteaminen.

2. Toiminnallisten ja ei-toiminnallisten tarpeiden määrittäminen käyttäjän näkökulmasta.

3. Ratkaisujen kehittäminen.

4. Ratkaisujen arvioiminen aikaisemmin määritettyjä tarpeita vasten.

Askelmia toistetaan niin kauan, että viimeisen askeleen arvioinnissa tullaan siihen tulokseen, että kehitetty järjestelmä vastaa tyydyttävästi aikaisemmissa askelmissa tehtyjä määritelmiä.

Käytettävyyden eri määritelmissä, kuten eri ISO-standardeissa (ISO 9126 & ISO 9421-11), Nielsenin [1993] ja Sharpin ja kollegoiden [Sharp & al, 2007], voidaan havaita eroja ja nämä erot tekevät eri määritelmien pohjalta tehtyjen tutkimuksien vertailun vaikeaksi. Seffah ja Metzker [2004] avaavat keskustelua artikkelissaan tästä ongelmasta. Yhdeksi esimerkiksi he nostavat standardien ISO 9421-

11 ja ISO 9126 opittavuuden määrittelyn eron. ISO 9421-11 määrittelee opittavuuden yksinkertaiseksi ”oppimiseen kuluva aika”-aliattribuutiksi. ISO 9126 puolestaan määrittelee opittavuuden itsenäiseksi laatutekijäksi, joka voidaan puolestaan jakaa useisiin tekijöihin kuten ymmärrettävään syötteeseen ja tulosteeseen, ohjeistettavuuteen ja viestinnälliseen valmiuteen. Seffah ja Metzker [2004] ovat oikeassa siinä, että ilman yhtenäisiä ja vakaita määritelmiä käytettävyystutkijoiden oppien seuraaminen on työlästä varsinaisille ohjelmistokehittäjille.

Yhtenä ongelmana he näkevät käytettävyystutkijoiden ja ohjelmistojen kehittäjien jakautumisen kahteen eri leiriin. Ensimmäisenä leirinä he näkevät ohjelmistokehittäjät, jotka ovat järjestelmän oikeat suunnittelijat ja toteuttajat. Ohjelmistokehittäjät pystyvät suunnittelemaan ja toteuttamaan luotettavia ja turvallisia järjestelmiä tehokkailla toiminnoilla. Ohjelmistokehittäjät jättävät käytettävyyden parantamisen käytettävyydestä vastaaville henkilöille. Toisena vastapuolena he näkevät edellä mainitut käytettävyystutkijat, jotka koostuvat psykologeista ja vuorovaikutteisuuden suunnittelijoista. He haluaisivat puolestaan suunnitella ja testata järjestelmän käyttöliittymän ja käytettävyyden loppukäyttäjien kanssa. Tämän jälkeen ensimmäisen leirin ohjelmistokehittäjät suunnittelevat järjestelmän tukemaan suunniteltua käyttöliittymää ja sen kautta suoritettavia tehtäviä. Tämän leiriytymisen vuoksi ohjelmistokehittäjien ja käytettävyystutkijoiden väliin syntyy muuri, joka ei edistä käytettävyyden toteuttamista järjestelmiin.

Toinen Seffah ja Metzkerin artikkelin esiin nostama ongelma on uskomus siitä, että käytettävyysihmisten tavoitettavuus tai saatavuus olisi helppoa järjestelmien kehityksen aikana. Tästä puolestaan seuraa, että käytettävyyden oppeja seurataan epämuodollisesti. Tämä ilmiö korostuu etenkin pienissä organisaatioissa, joissa vuorovaikutteisia järjestelmiä kehittävät henkilöt, joilla ei ole vuorovaikutteisesta tekniikasta tarpeellista kokemusta. Näissä pienissä organisaatioissa ei välttämättä ole omaa käytettävyystutkijaa, mutta tämä ongelma voitaisiin kiertää ulkopuolisen käytettävyyskonsultin avulla. Tämän puolestaan monesti voivat estää kustannustekijät.

3.3 Käytettävyys ja tehokkuus

Käytettävyyden määritelmässä yhtenä osana oli tehokkuus, jonka määrittelyt poikkeavat pienin vivahtein toisistaan. Nielsen [1993] viittaa tehokkuudella ta-

saisena pysyvään suoritustasoon oppimiskäyrän tasoittumisen jälkeen. Tällä viittauksella tarkoitetaan kokemattoman käyttäjän alhaista suoritustasoa oppimiskäyrän alkupäässä, jolloin käyrä on nouseva. Kokeneeksi käyttäjäksi voidaan kutsua henkilöä, joka on ohittanut oppimiskäyrän alkupään nousuosan ja siirtynyt käyrän tasaisena pysyvään osaan. Se, miten nopeasti kokematon käyttäjä saavuttaa kokeneen käyttäjän tason, riippuu käytettävän järjestelmän asettamista vaatimuksista käyttäjää kohtaan. Aikaisemmin mainittujen Googlen hakukoneen (kuva 2) ja Rohde & Schwarzin CMU200 (kuva 1) kohdalla oppimiskäyrän tasaisen vaiheen saavuttamisen aikaero on useita vuosia, mikäli käyttäjällä ei ole telekommunikaatiosta pohjatietoa.

Sharp kollegoineen [Sharp & al., 2007] toisaalta viittaavat tehokkuudella siihen, miten suunniteltu järjestelmä helpottaa käyttäjää tehtävien suorittamisessa. Tämänkin viittauksen taakse voidaan kuvitella kunkin järjestelmän oppimiskäyrän vaikutus. Aikaisemmasta esimerkistä johdettuna Googlen hakukone ja Rohde & Schwarzin tehokkuus käyttäjän tehtävän suorittamisen näkökulmasta voi olla sama, kunhan käyttäjä on näiden kahden järjestelmän oppimiskäyrillä tasaisella osiolla.

Frokjaer ja kollegat [2000] viittaavat rutiinitehtävien hyvään tehokkuuteen, kun suoritetut tehtävät koostuvat hyödyllisistä hyvin suoritetuista osatehtävistä. Hyvän tehokkaan järjestelmän kohdalla tämä tarkoittaa sitä, että käyttäjä kykenee suorittamaan saman tehtävän useita kertoja peräkkäin hyvin ilman suurta laadullista tai ajallista vaihtelua. Tätä viittausta tehokkuuteen voidaan käyttää myös käytettävyyden indikaattorina, koska rutiinitehtävien toistojen ajat ovat vertailukelpoisia. Esimerkkinä rutiinitehtävästä voidaan pitää yliopiston opiskelijan kirjan varausta tiedekunnan ja kaupungin kirjastosta käyttäen kirjastojen verkkosivuja. Opiskelijalle kirjan varaus kirjaston verkkosivujen kautta muodostuu pienemmistä osatekijöistä, kuten verkkosivulla navigoinnista, kirjan hakemisesta, järjestelmään kirjautumisesta ja itse kirjan varaamisesta. Useamman vuoden yliopistossa opiskelleelle näiden vaiheiden voisi olettaa olevan rutinoituneita prosesseja. Näin ollen Frokjaerin ja kollegoiden tehokkuuden määrittelyä mukaillen voidaan suorittaa vertailu tiedekunnan ja kaupungin verkkosivujen käytettävyydestä, koska kirjan varauksen kaltaisesta rutiinitehtävästä saadaan vertailukelpoinen indikaattori.

4 Käytettävyyden luoma tehokkuus

Käytettävyyden luoman tehokkuuden yksiselitteinen mittaaminen järjestelmästä ei ole helppo tehtävä. Tehtävän suorittamiseen kuluvaa aikaa mittaamalla voitaisiin saada vertailukelpoista tietoa käytettävyyden parantumisen tuomasta tehokkuudesta, mutta täysin uuden järjestelmän kohdalla tämä ei ole mahdollista vertailukelpoisen tiedon puuttuessa. Eräänä ratkaisuna uuden järjestelmän käytettävyyden tehokkuuden arvioimisessa pitäisin kahden eri käyttöliittymätoteutuksen kehittämistä uuteen järjestelmään erillisinä prosesseina. Testausvaiheessa uuden järjestelmän käytettävyyden tehokkuudesta voitaisiin saada hieman tietoa vertaamalla uuden järjestelmän kahta erilaista käyttöliittymää toisiinsa. Tämä voi olla hankala toteuttaa resurssien puolesta.

Käyttäjän ja järjestelmän luoman yhdistelmän tehokkuus voidaan arvioida muustakin näkökulmasta kuin pelkän tehtävän suoritusajan perusteella. Esimerkiksi yrityksen johtoa kiinnostaa tietää uutta järjestelmää hankittaessa uuden järjestelmän opiskeluun kuluva aika, eli uuden järjestelmän oppimiskäyrä. Toisena erilaisena esimerkkinä voidaan mainita viihdejärjestelmien, kuten pelikonsolin tai television, käytettävyyden tehokkuuden mittaaminen. Viihdejärjestelmissä ymmärrettävästi järjestelmän käyttötarkoitus ei ole auttaa käyttäjää suorittautumaan tehtävästä mahdollisimman nopeasti ajallisesti, vaan auttaa käyttäjää kuluttamaan käyttäjän haluama määrä aikaa. Tietokone- tai konsolipelissä tämä voisi merkitä esimerkiksi hyvin toteutettua pelitilanteen tallentamista. Pelien käytettävyyden tehokkuusvertailu voitaisiin suorittaa vaikka 1980-luvulla tehtyjen pelien ja nykyaikaisten pelien välillä. 1980-luvulla käytössä olleen tekniikan vuoksi pelitilanteita ei usein pystynyt tallentamaan ollenkaan. Peleissä saattoi olla eri kenttiä tai tasoja, joille päästessään pelaaja sai koodin, jonka avulla pelaaja pystyi aloittamaan pelin kentän tai tason alusta. Kentän tai tason alku toimi toisin sanoen eräänlaisena tallennuspisteenä. Tällä mekanismilla pelaaja ei ihan täysin pysty päättämään käyttämäänsä peliaikaa ilman pelitilanteen menetystä. Nykypeleissä on puolestaan usein parantunut pelitilanteen tallennus, jolloin käyttäjä voi pysäyttää pelin haluamansa kohtaan ja jatkaa samasta tilanteesta myöhemmin uudelleen. Tällaisessa tapauksessa voitaisiin mitata viihdejärjestelmän tehokkuutta vertaamalla sitä, kuinka hyvin käyttäjä pystyi käyttämään juuri haluamansa määrän aikaa pelaamiseen.

Käyttäjän mielikuva järjestelmän tehokkuudesta on varsinkin markkinoinnin näkökulmasta tärkeä tekijä. Käyttäjän näkökulmasta tutkittu järjestelmän tehokkuus on tuottanut varsin mielenkiintoisia tutkimustuloksia Raidan ja Oulasvirran tekemässä tutkimuksessa [2010]. Tutkimuksessa järjestelmästä annettiin koehenkilöille ennen varsinaista testiä joko positiivinen, negatiivinen tai neutraali mielikuva. Neutraalin mielikuvan tapauksessa käyttäjää ei pohjustettu millään tiedolla ennen testiä. Tämän pohjustuksen jälkeen koehenkilöt suorittivat annetut tehtävät. Tehtävien suoritusta valvottiin erilaisin menetelmin ja tehtävien jälkeen koehenkilöt täyttivät kyselykaavakkeen. Pohjustuksen vaikutus tehtävän kuormittavuuteen NASA-TLX-menetelmällä mitattuna osoittaa pienellä marginaalilla, että positiivisen mielikuvan järjestelmästä saaneilla tehtäväkuormitus oli pienempi helpoissa tehtävissä. Negatiivisen ja neutraalin mielikuvan saaneiden lähtötila tehtävänkuormittavuuden osalta oli testin mukaan sama. Koetehtävän vaikeutuessa positiivisen mielikuvan saaneen tehtäväkuormitus kasvaa samaa rataa neutraalin mielikuvan omaavien kanssa. Negatiivisen mielikuvan saaneilla tehtävänkuormituksen nousu on tuloksien mukaan selvästi loivempaa, mikä on sinällään yllätyksellistä. Toisaalta sananlasku “pessimisti ei pety koskaan” sopii selittämään tätä ilmiötä. Sama ilmiö on näkyvillä myös testin jälkeen tehdyssä käytettävyyskyselyssä, jossa positiivisen mielikuvan saaneet arvioivat järjestelmän käytettävyyden selvästi korkeammaksi kuin negatiivisen ja neutraalin mielikuvan saaneet. Käytettävyysarvioiden pisteiden lasku tehtävien vaikeutumisen myötä noudattaa tehtäväkuormituksen teemaa. Eli negatiivisen mielikuvan saaneiden arviointipisteiden arvo ei laske niin jyrkästi kuin positiivisen ja neutraalin mielikuvan saaneilla.

Kuten aikaisemmassa luvussa todettiin, käytettävyyden määritelmien monimuotoisuuskin aiheuttaa omat ongelmansa erilaisille käytettävyyttä mittaaville testeille ja niistä saaduille tuloksille. Frokjaerin ja kollegoiden [2000] tutkimuksessa tultiin siihen tulokseen, että tehokkuuden, hyödyllisyyden ja tyytyväisyyden välisiä relaatioita ei ymmärretä riittävän hyvin. Tämä tarkoittaa heidän mukaansa sitä, että tehokkuutta, hyödyllisyyttä tai tyytyväisyyttä tulisi tarkastella yksittäisinä ja riippumattomina käytettävyyden tekijöinä.

5 Järjestelmän arviointi GOMS-mallilla

Käytettävyyden luoman tehokkuuden mittaaminen järjestelmästä ei ole helppoa, mutta sen arviointi tietyillä menetelmillä on mahdollista. Eräänä menetelmänä käytettävyyden ja sen luoman tehokkuuden mittaamiseen insinöörityökaluista voidaan käyttää GOMS-mallia (Goals, Operators, Methods and Selection rules). GOMS-mallilla tehdyt analyysit sopivat tilanteisiin, joissa käyttäjillä voidaan olettaa olevan pohjatietoa suoritettavasta tehtävästä [Carroll, 2003, s. 60]. Myöhemmin tarkasteltavissa käyttöliittymissä tämän mallin voidaan olettaa toimivan kohtuullisesti sen jälkeen, kun käyttäjä on hieman tutustunut työkaluun. Tällöin tehokkuutta tarkastaltessa voidaan keskittyä tehtävän suorittamiseen.

5.1 GOMS-mallin yleiskuvaus

GOMS-lyhenne muodostuu sanoista Goals (tavoitteet, maalit), Operators (operaattorit), Methods (menetelmät) ja Selection rules (valintasäännöt). Nämä komponentit ovat GOMS-mallin peruspilareita, joiden varaan kohteesta tehtävä malli tukeutuu. Ajatusmaailma GOMS-mallissa lähtee siitä, että käyttäjällä on jokin tavoite, mihin hän pyrkii käyttäessään tietokonetta tai jotain muuta laitetta. Tämä maali tai tavoite taas on saavutettavissa, kun käyttäjä suorittaa yhden tai useamman operaattorin, eli alkeistehtävän, joka vie käyttäjän asettamaansa tavoitteeseen. Mikäli operaattorit ryhmittäytyvät erillisiin kokonaisuuksiin, joiden avulla voidaan saavuttaa jokin selkeä välitavoite, niin ne muodostavat silloin erillisen menetelmän. Valintasääntöjä puolestaan tarvitaan, mikäli samaan käyttäjän asettamaan tavoitteeseen on mahdollista päästä eri menetelmiä pitkin. Hyvin yksinkertaistettuna esimerkkinä voidaan mainita erämaassa vaelluksella henkilö, joka saapuu joen luo, jonka ylitse hänen täytyy päästä matkaa jatkaakseen. Tässä tilanteessa edellä mainitut GOMS-mallin peruspilarit karkeasti määriteltynä ovat seuraavat:

- Tavoite: Joen ylitse pääseminen.
- Operaattorit: mm. askeleen ottaminen, rakentaminen tai etsiminen.
- Menetelmät: mm. kahlaaminen, uiminen, lautan rakennus tai valmiin sillan etsiminen.
- Valintasäännöt: mm. kuivana säilyminen, käytettävä aika lautan rakentamiseen tai valmiin sillan kautta kiertämiseen.

GOMS-mallia voidaan avata myös Carrollin kirjan tavoin tekstineditointitehtävän [Carroll, 2003, s. 59] avulla. Tehtävässä kuvitteellisen käyttäjän tulee siirtää lauseesta “The fox jumps over the lazy quick brown dog.” sanapari “quick brown” sanan “fox” eteen.

5.1.1 Tavoitteet (Goals)

Tavoitteet GOMS-mallin yhteydessä voidaan mieltää päämääriksi, joita kohti käyttäjä pyrkii. Käyttäjän pyrkiessä kohti päämääräänsä, eli tavoitettansa, matkalla saattaa olla tunnistettavia osatavoitteita. Tässä tapauksessa päätason tavoite saavutetaan, kun kaikki nämä osatavoitteet on suoritettu onnistuneesti. Tavoitteet osatavoitteineen voidaan järjestää hierarkkisesti selkeyden vuoksi. Ylempänä olleessa vaellustapauksessa tavoitteena vaeltajalla oli päästä joen yli. Osatavoitteena samassa tapauksessa voisi olla esimerkiksi valmiin sillan löytäminen kartalta tai lautan rakentamisen valmistuminen. Tekstieditointiesimerkissä osatavoitteeksi voidaan mieltää esimerkiksi kursorin siirto haluttuun kohtaan.

5.1.2 Operaattorit (Operators)

Operaattori-sanalla tarkoitetaan toimia, joita käyttäjän sallitaan tehdä käyttöliittymässä tai käyttötilanteessa. Toimia voisi pitää GOMS-mallin juuritasona, joka tarjoaa mallille liityntäkohdan reaalimaailmaan. Operaattorit voidaan jakaa havaittaviin ja näkymättömiin yksikköihin. Havaittavat operaattorit ovat usein konkreettisia, joiden mittaaminen ulkoisilla välineillä tai menetelmillä on mahdollista. Näkymättöminä operaattoreina voidaan pitää esimerkiksi henkilön valintaan johtavaa ajatusprosessin osia, joiden havaitseminen tämän hetkisen teknologian avulla on työlästä tai täysin mahdotonta. Operaattorit voivat liittyä mm. seuraaviin termeihin:

- Motorisiin, jotka edustavat liikettä. Motorinen operaattori on yleensä havaittava.
- Havainnointiin tai aistittaviin, jotka yksittäin ovat yleensä näkymättömiä.
- Kognitiivisiin, jotka liittyvät ajatteluun tai tietämiseen. Nämä mielletään usein myös näkymättömiksi.

Operaattori voi olla myös yhdistelmiä näistä ylläolevista, mikäli operaattoreita ei viedä liian syvälle. Mallin tarkkuuden yhtenä tekijänä on operaattoreiden määrittelyn yksityiskohtaisuus. Hyvin yksityiskohtaisien operaattorien käyttö on hyvin työlästä ja aikaa vievää. Tästä johtuen mallinsuunnittelun alkuvaiheessa on hyvä määrittää tarkoitukseen riittävä taso operaattoreille.

Komentorivipohjaisessa käyttöliittymässä yksittäinen toimi voi olla esimerkiksi yksi komento parametreineen, jotka käyttäjä syöttää näppäimistöllä. Graafisissa käyttöliittymissä käyttäjällä sallitut toimet tehtävän suoritukseen voivat vaihdella vaihtoehtojen valitsemisesta valikoista yksittäisen napin painalluksen kautta suoraan kohteen manipulointiin. Nykyajan tekniikka on mahdollistanut myös toimet muun muassa eleillä, puheella ja silmän liikkeillä. Toimia voidaan yleisesti kuvata hyvin usealla eri abstraktion tasolla, mutta useimmat GOMS-mallit kohdistavat toimet hyvin konkreettiselle tasolle selkeyden vuoksi [Carroll, 2003, s. 59]. Aikaisemmin mainitussa tekstieditointitehtävässä käyttäjälle sallitut toimet olisivat esimerkiksi näppäimistöllä tekstin maalaus ja siirto tai hiirellä tekstin maalaus ja valikkojen leikkaa&liimaa-toimintojen avulla tekstin siirto. Taulutietokoneiden, joissa ei ole erillistä näppäimistöä, käyttäjällä on käytössään kosketusnäytölle määritellyt toimet tekstin valitsemiseen ja siirtoon. Vaellusesimerkissä motorisina operaattoreina toimivat esimerkiksi askeleen ottaminen tai lautan rakentamiseen tarvittavan oksan nosto. Kognitiivisena operaattorina puolestaan henkilöllä on mahdollisen tietouden hyödyntäminen ylitettävänä olevasta joesta parasta ylitystapaa valittaessa.

5.1.3 Metodit (Methods)

Metodeilla tarkoitetaan opittuja erilaisista toimista koostuvia sekvenssejä, joilla voidaan saavuttaa tavoite. Aikaisemmin mainitun esimerkkitehtävän yhteydessä graafinen tekstieditori sisältää seuraavat metodit:

- Merkitse siirrettävä teksti.
- Paina näppäimistöltä ctrl-x leikataksesi tekstin.
- Siirrä hiiren kursori haluttuun kohtaan.
- Paina hiiren nappia halutussa kohdassa.
- Paina näppäimistöltä ctrl-v liimataksesi teksin.

On helppoa löytää ylläolevasta listasta konkreettiselle tasolle menevät näppäinpainallustason toimet sekä välitavoitteet. On tärkeää myös huomata, että sama tekstieditori voi sisältää useamman eri tavan saavuttaa välitavoite. Esimerkkinä voidaan pitää vaikka tekstin leikkaamista “ctrl-x”-painalluksena. Vaihtoehtoisena toimena näppäinyhdistelmälle käyttäjä voisi valita tekstieditorin menusta leikkaatoiminnon ja päästä siten samaan tavoitteeseen, eli tekstin leikkaamiseen.

Vaellusesimerkissä taas puolestaan henkilöllä oli joen toiselle puolelle pääsemisessä käytettävissään kolme eri menetelmää. Jokaisesta menetelmästä on selkeästi havaittavissa toimintaketjut, joista menetelmä koostuu. Esimerkiksi valmista siltaa etsiessä käyttäjä voi tehdä seuraavat toimet:

- Määritä sijainti kartalta.
- Etsi lähin tai nopeimmin saavutettavissa oleva silta, jonka avulla joen yli pääsee.
- Määritä suunta, mihin täytyy lähteä sillan luokse pääsemiseksi.

5.1.4 Valintaehdot (Selection Rules)

Valintaehdot on GOMS-mallin viimeinen komponentti, jota tarvitaan tilanteissa, joissa saman tavoitteen saavuttamiseen voidaan käyttää useampaa eri metodia. Edellisessä Metodi-alikappaleessa mainittiin tekstin leikkaamiselle kaksi eri metodia, joilla päästään samaan lopputulokseen ja näiden kahden vaihtoehdon lisäksi

on olemassa vielä lisää vaihtoehtoja esimerkiksi kosketusnäytöllisissä käyttöliittymissä. Valintaehdot pohjautuvat käyttäjän omiin mieltymyksiin ja toimintatapoihin tehdä asia. Valinnat voivat vaihdella myös tehtävän mukaan. Esimerkiksi kokonaisen rivin leikkaaminen ja liimaaminen voi olla mielekkäämpää tehdä näppäimistöltä hiirellä työskentelyn sijaan toisin kuin aikaisemmin mainitussa esimerkkitehtävässä, jossa siirrettiin lauseen keskellä olevia sanoja kokonaisen rivin sijaan. Valintavaihtoehdot vaihtelevat myös käyttäjien välillä. Esimerkiksi kokeneempi sovelluksen käyttäjä todennäköisesti tietää tehtävän suorittamista nopeuttavia oikopolkuja, jotka eivät satunnaiselle käyttäjälle ole niin selviä. Yhtenä esimerkkinä toimii näppäimistöltä tehtävät valinnat, kuten leikkaa “ctrl-x”-komento ja liimaa “ctrl-v”-komento. Käyttäjä, joka ei näitä tiedä, valitsee niiden sijaan valikoista löytyvät vastineet edellisille komennoille. Vaellusesimerkissä valintasääntöjen hahmottaminen voi olla helpompaa. Esimerkin mukaisessa tilanteessa valintasäännöt voisivat olla esimerkiksi seuraavat:

- Ylitä joki siltaa pitkin, mikäli silta löytyy riittävän läheltä nykyistä sijaintia.
- Ylitä joki kahlaten, mikäli joki ei ole liian syvä.
- Rakenna lautta ylitystä varten, mikäli sopivaa rakennusmateriaalia on saatavilla.
- Ylitä joki uiden, mikäli muut vaihtoehdot ovat mahdottomia tai liikaa aikaa vieviä.

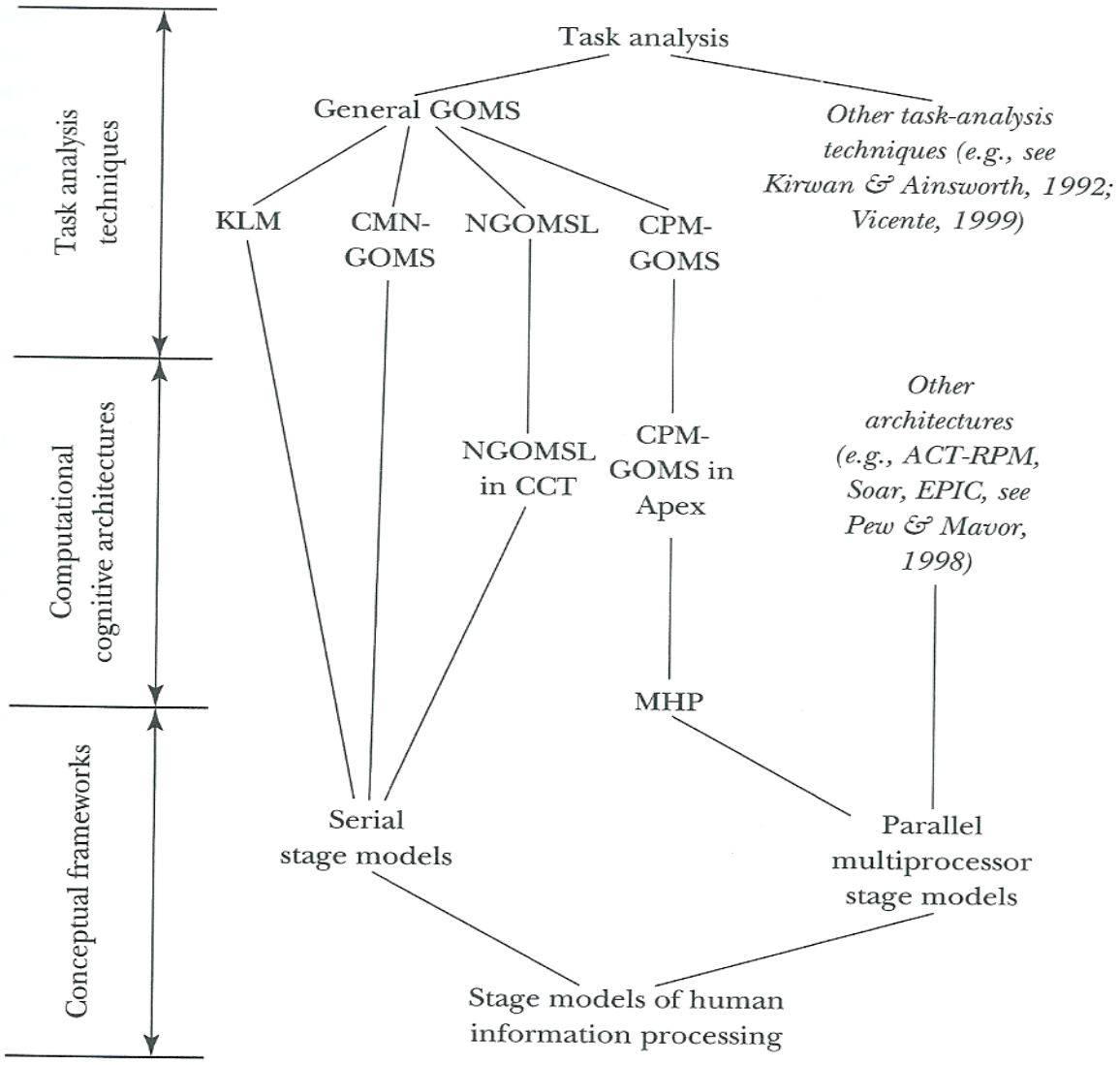
Näiden yllämainittujen valintasääntöjen välillä ei ole suurta eroa, jos kriteerinä pidettäisiin pelkästään tavoitteeseen pääsyä, eli joen toiselle puolelle pääsyä. Käytännön vaellustilanteessa valintasäännöillä on hyvin konkreettiset vaikutukset joen ylitykseen kuluvaan aikaan ja vaeltajan tilaan joen ylityksen jälkeen. Tässä kohtaa on helppo ymmärtää myös ympäristön vaikutus päätöksen tekoon. Tästä esimerkkinä voidaan mainita joen ylittäminen kahlaamalla lämpimällä kesäkelillä tai kylmempänä syyspäivänä. Vaeltajan päätöksentekoon ulkoisena parametrina tulee ympäristön lämpötila ja varusteiden kuivattamisen mahdollisuus vastarannalla.

5.2 GOMS-mallin rajoitukset

GOMS-mallilla on kolme tärkeää rajoitetta [Carroll, 2003], jotka on hyvä tiedostaa ennen mallin käytön harkitsemista. Ensimmäinen näistä on se, että analysoitavan tehtävän tulee olla hyvin ymmärretty tarvittavien toimenpiteiden puolesta. Tämän saavuttamiseksi tehtävän suorittamisen kannalta olennainen pohjatieto tulee olla myös hallussa. Näistä johtuen GOMS-malli on proseduraalinen. Toinen GOMS-mallin rajoitus muodostuu sen esitystavasta, joka koostuu proseduraalisesta tietämyksestä tai muodollisista ohjeista. Tästä johtuen mallin esitystapa voidaan mieltää taidokasta käytöstä tukevaksi tietyn kaavan mukaan. Tästä johtuen mallin esitystapa ei sovellu kaikkiin reaalielämän tilanteisiin, joissa käyttäjä voi lähestyä ongelman ratkaisua intuition avulla tai kokeiluperiaatteella. Kolmantena rajoituksena on mallin luonteesta johtuen se, että suunnittelijan tulee aloittaa määrittämällä kaikki päätason tavoitteet, joita GOMS-malli itsessään ei tarjoa suunnittelijalle suoraan. Päätason tavoitteet tulevat mallin ulkopuolelta esimerkiksi potentiaalisten käyttäjien haastattelujen perusteella tehtävistä tehtäväanalyyseistä. Tässä rajoitteessa piilee myös GOMS-mallin heikkous puutteellisen tai virheellisen tavoite määrittelyn suhteen. Esimerkiksi suunnittelijan ja haastateltavan välisen kommunikaation ongelma saattaa johtaa virheelliseen päätason tavoitteenmääritykseen ja tämän jälkeen mallia rakennettaessa kyseisen päätason tavoitteen alaiset proseduurit ja metodit voivat johtaa ihan eri tilaan, kuin mitä haastateltava yritti haastattelutilanteessa suunnittelijalle selittää.

5.3 GOMS-mallin eri versiot

GOMS-malli alkuperäisen muotonsa jälkeen on saanut viereensä erilaisia muunnoksia. Seuraavasta Carrollin kirjan kuvasta voidaan nähdä kirjan kirjoittamishetken tilanne GOMS-sukupuusta.



Kuva 3: GOMS-sukupuu [Carroll, 2003, s. 57].

5.3.1 KLM

KLM-mallia (Keystroke-Level Model) voidaan pitää GOMS-sukupuun yksinkertaisimpana muotona ja sen soveltaminen vaatii tarkkaa ja huolellista valmistelua analysoijalta [Card, Moran & Newell, 1980a]. Arkkitehtuuri KLM-mallissa perustuu kognitiivisuuteen, jonka avulla jäljitellään käyttäjän tapaa käsitellä informaatiota yksittäisinä tapahtumina sarjamuotoisesti. Oikein käytettynä KLM-mallilla voidaan saavuttaa tarkkoja analyysejä kohteesta nopeasti. Tämä on mahdolista, mikäli analysoitavasta kohteesta voidaan määrittää selkeä tavoite, joka on saavutettavissa sarjalla operaattoreita. Heikkoutena KLM-mallilla on sen käytön hankaluus laajemmissa kokonaisuuksissa, joissa käyttäjän tekemien toimien ennustettavuus on heikko tai täysin mahdoton. Alun perin KLM-mallissa oli kuusi eri operaattoria, joiden avulla koneella tehtävää tehtävää voidaan analysoida. Nämä operaattorit ovat seuraavat:

Operaattori	Selite
K (Key)	Näppäimen painallus.
P (Point)	Hiiren kursorin siirtäminen näytöllä olevan kohteen päälle.
H (Hand)	Käsien vienti näppäimistön näppäinten päälle tai käsien liikuttaminen jonkin toisen kontrollilaitteen päälle.
D (Draw)	Viivan piirto ruudukolla.
M (Mental)	Henkinen valmistautuminen johonkin tiettyyn yksittäiseen toimintaan tai sarjaan operaattoreita.
R (Response)	Järjestelmän vasteaika, jonka käyttäjä joutuu odottamaan antamansa syötteen jälkeen siihen pisteeseen, että järjestelmä antaa jonkin palautteen käyttäjän antamaan syötteeseen.

Taulukko 1: KLM-mallin operaattorien esittely [Card & al, 1980a].

Analyysin suoritusta varten jokaiselle operaattorille on määritetty arvioitu suoritusaika ja näiden aikojen pohjalta kokonaisaika suorituksen kestolle voidaan laskea.

KLM-mallissa M-operaattorille oli alussa määritelty viisi eri heuristiikkasääntöä henkiselle valmistautumiselle. Nämä viisi heuristiikkasääntöä ovat seuraavat:

- Sääntö 0. Alustava sijoituskandidaatti M:lle. Operaattori M sijoitetaan jokaisen P-operaattorin, joka valitsee jotain, eteen. Poikkeuksena valinnat, jotka toimivat käskyn argumenttina.

- Sääntö 1. Ei toivottujen M:ien poisto. M poistetaan P-operaattorin edestä, mikäli P-operaattori on täysin odotettavissa edellisten toimien pohjalta.

- Sääntö 2. M-operaattorien poisto kognitiivisista yksiköistä (esimerkiksi yksi sana). Kaikki muut M-operaattorit ensimmäistä lukuun ottamatta poistetaan, mikäli ne esiintyvät samassa kognitiivisessa yksikössä.

- Sääntö 3. M-operaattorin poisto sarjan päättäjän edestä. Mikäli K on tarpeeton päättäjä kognitiiviselle sarjalle, niin M poistetaan sen edestä.

- Sääntö 4. M-operaattorien poisto, mikäli ne ovat käskyjen päättäjiä. Jos erottimena toimiva K-operaattori on osa jonoa, niin M-operaattori ennen sitä poistetaan.

- Sääntö 5. Päällekkäisten M-operaattorien poisto. M-operaattorin aikaa ei lasketa tapauksissa, joissa se on päällekkäin R-operaattorin kanssa esimerkiksi tietokoneen vasteen odottelun yhteydessä.

Aikaisemmin mainittu esimerkin alkuosa KLM-mallina voisi näyttää seuraavalta:

Tehtävän kuvaus	Operaattori	Kesto (sekunteja)
Valmistaudu henkisesti	M	1,35
Siirrä kursori sanan “quick” alkuun	P	1,1
Paina hiiren nappi pohjaan	K	0,2
Siirrä osoitin sanan “brown” loppuun	P	1,1
Valmistaudu henkisesti	M	1,35
Siirrä osoitin Muokkaa-valikon päälle	P	1,1
Klikkaa hiirellä	K	0,2
Siirrä osoitin Leikkaa-kohdan päälle	P	1,1
Klikkaa hiirellä	K	0,2
Valmistaudu henkisesti	M	1,35
Siirrä osoitin sanan “fox” eteen	P	1,1
Klikkaa hiirellä	K	0,2
Valmistaudu henkisesti	M	1,35
Siirrä osoitin taas Muokkaa-valikon päälle	P	1,1
Klikkaa hiirellä	K	0,2
Valmistaudu henkisesti	M	1,35
Siirrä osoiten Liimaa-kohdan päälle	P	1,1
Klikkaa hiirellä	K	0,2

Taulukko 2: Esimerkin yksi KLM-mallin mukainen esitystapa.

Kuten aikaisemmin todettiin, KLM-mallin käyttö sopii tietynlaisiin tilanteisiin hyvin. Yhtenä tälläisena esimerkkinä Kieras [2005, s. 9] esittelee tiedoston poiston Mac OS:n alkuperäisessä suunnitelman mukaisesti ja vertaa sitä kokeneen käyttäjän tekemään tiedoston poistoon komento-näppäimen avulla. Mac OS:n alkuperäisen suunnitelman esimerkin analysointia varten joudutaan tekemään seuraavat oletukset:

- Tiedostojen ikonin raahaminen roskakoriin on yleinen käytäntö tiedostojen poistoon.

- Kokenut käyttäjä ajattelee halutun ikonin valitsemista ja raahaamista yhtenä operaationa. Halutun ikonin löytäminen kuitenkin tulee pitää omana operaationa, koska se on erilainen jokaisella suorituskerralla.

- Roskakorin paikannusta näytöltä ei tarvita erillisenä operaationa, koska sen korvaa roskakorin osoittaminen hiiren kursorilla.

- Roskakoriin osumisen tarkistamista ei tarvita erillisenä operaationa, koska hiiren kursorilla roskakorin osoittaminen korvaa sen.

- Roskakorin tilan tarkistamista ei suoriteta.

Kieras määritteli esimerkin tilanteeseen seuraavat operaattorit [2005, s. 8]:

- K - Näppäinpainallus (0.12-1.2 s), käytä 0.28 s tavalliselle käyttäjälle. Näppäimen tai painikkeen painallus näppäimistöllä.

- B - Hiirennapin painallus tai vapautus (0.1 s, klikkaus 0.2 s). Tavallisen käyttäjän tapauksessa tätä voidaan pitää hyvin harjoiteltuna ja yksinkertaisena operaationa.

- P - Kohteen osoittaminen näytöllä hiiren kursorilla. Tyypillisesti vaihtelee 0.8 - 1.5 s välillä. Keskimääräinen aika tekstin editoinnissa on 1.1 s.

- H - Käsien siirto näppäimistön tai hiiren päälle (0.4 s).

- W - Järjestelmän palautteen odottaminen. Tähän operaattoriin kuluva aika tulee määrittää käytössä olevasta järjestelmästä.

- M - Henkinen valmistautuminen. Kuvaa taukoa eri operaattoreiden suorituksen välissä. Kokemattomat käyttäjät vaativat näitä enemmän kuin kokeneemmat käyttäjät. Arviot kestosta vaihtelevat 0.6 s - 1.35 s välillä ja 1.2 sekuntia pidetään hyvänä keskiarvona tälle.

Esimerkin tehtävänä on löytää tuhottavan tiedoston ikoni ja raahata se roskakoriin. Esimerkki koostuu seuraavasta operaattorien muodostamasta sekvenssistä:

Tehtävän kuvaus	Operaattori
Aloita henkisesti tiedoston poisto-operaatio	M
Etsi haluttu kohde	M
Osoita tiedoston ikonia	P
Paina ja pidä pohjassa hiiren nappia	B
Raahaa ikoni roskakorin ikonin päälle	P
Vapauta hiiren nappi	B
Siirrä kursori alkuperäisen ikkunan päälle	P

Taulukko 3: Esimerkin operaattorisekvenssi.

Ylläolevan operaattorisekvenssin suoritukseen kuluu aikaa seuraavasti: 3P + 2B +2M = 5.9 s.
Kokeneemman käyttäjän tehdessä tiedoston poiston komento-näppäimellä Kieras olettaa, että käyttäjä käyttää ainoastaan oikeaa kättä hiiren ja näppäimistön käyttöön. Oletuksena on myös, että oikea käsi aloittaa ja lopettaa hiiren päällä. Tehtävän suoritus komento-näppäimen avulla koostuu halutun tiedoston ikonin valitsemisesta ja komento-näppäimen painalluksesta. Tämä muodostaa seuraavanlaisen listan operaattoreista:

Tehtävänä kuvaus	Operaattori
Aloita henkisesti tiedoston poisto-operaatio.	M
Etsi haluttu kohde.	M
Osoita tiedoston ikonia.	P
Klikkaa hiiren nappia.	BB
Siirrä käsi näppäimistön päälle.	H
Näpäytä komento-näppäintä.	KK
Siirrä käsi takaisin hiirelle.	H

Taulukko 4: Operaattorilistaus.

Tähän listaukseen kuluu aikaa seuraavasti: P + 2B + 2H + 2K + 2M = 5.06 s. Aika on vain hieman parempi alkuperäiseen suunnitelmaan nähden käden siirron vuoksi. Jos oletettaisiin Kieraksen esimerkistä poiketen, että käyttäjällä olisi lähtötilanteessa vasen käsi näppäimistöllä, niin suoritusaika lyhenisi kahden H-operaattorin verran. Tämän esimerkin vertailu kuvastaa hyvin KLM-mallin käyttöä osoittamalla tehtyjen oletusten vaikutuksen mallin tuottamaan tulokseen.

5.3.2 CMN-GOMS

CMN-GOMS-malli [Card & al., 1980b] on Cardin, Morganin ja Newellin tarkentama kuvaus yleisestä GOMS-mallista. Ero alkuperäiseen malliin tulee hieman yksityiskohtaisemmasta tavoitehierarkiasta, toimintojen tarkasta sekventiaalisesta suoritusjärjestyksestä ja metodien pseudokoodimaisesta esitystavasta [Carroll, 2003, s. 75]. Osin CMN-GOMS-malli muistuttaa hieman Model Human Processor (MHP) -ajattelua, jossa luodaan analogia ihmisen ja koneen prosessoinnin ja muistivarastojen välille. Tämän analogian avulla ihmisen toimintaa voidaan mallintaa. CMN-GOMS-malli poikkeaa siitä periaatteellisesti sekventiaalisen ajojärjestyksen osalta. Cardin, Moranin ja Newellin GOMS-mallin kehittämisessä on ollut kantavana ajatuksena lähteä purkamaan ylätason tavoitteita eri vaiheissa niin pitkälle, kunnes haluttu tarkkuustaso on saavutettu. Alimmalla tasolla CMN-GOMS-mallin tuottama lista tarvittavista toimista muistuttaa hyvin pitkälle KLM-mallin mukaista listausta, mutta poikkeaa siitä kuitenkin määrittelemällä selkeät välitavoitteet ja tehtävät valinnat eri välitavoitteiden suhteen. Eroa muodostuu listattujen askelten luonteesta. Siinä, missä KLM-mallin askel on selkeästi havaittavissa ulkoisesti, GMN-GOMS-mallin askel on abstraktimpi. Esimerkkinä tästä käy aiemmin mainitun tekstinmuokkauksen vertaaminen KLM- ja CMN-GOMS-malleilla.

Lisäksi erona KLM-malliin CMN-GOMS-mallin tulos on ohjelmallisessa muodossa, jolloin se on analysoitavissa ja ajettavissa vaivatta. Tästä seuraa etu KLM-malliin nähden, koska analysoijan ei tarvitse sanella tarkkaa listausta eri tilanteiden ennusteiden esiin saamiseksi. CMN-GOMS-mallissa tehtäväkohtaista listausta ei tarvitse tehdä, koska tavoitteet voidaan saavuttaa mallin luomien välitavoitehaarojen mukaan ja näiden perusteella voidaan määrittää ennusteet. Ero KLM-malliin käy hyvin selväksi Carrollin kirjan CMN-GOMS-mallin mukaisesta listauksesta tekstin siirtoesimerkistä.

Kuva 4: CMN-GOMS-mallin mukainen esitys tekstin siirtoesimerkistä [Carroll, 2003, s. 77].

5.3.3 NGOMSL-malli

NGOMSL-malli (Natural GOMS Language) GOMS-sukupuussa on kaikkein eniten luontaisen kielen kaltainen ilmaisuasultaan ja rakenteeltaan [Kieras, 1988]. Yksi kantavista ideoista NGOMSL-mallissa on sen ohjelmamuotoisuus, jonka avulla voidaan kuvata korkean tason tavoitteita ja niihin tarvittavia metodeja selkeillä rakenteilla. Tästä johtuen NGOSML-mallia voi pitää käytännöllisenä, koska se tarjoaa tarkan proseduurin GOMS-mallin luomiseen kohteesta. NGOMSL-mallin juuret löytyvät CCT:stä (*Cognitive Complexity Theory*), jonka korkean tason notaatioksi NGOSML-malli luotiin [Kieras & Polson, 1985]. CCT:ssä jonkin tietyn laitteen käyttöön tarvittava tiedon laatu, määrä ja rakenne kuvataan tarkkaan. Kuvattu tieto voidaan jakaa kahteen eri tyyppiin:

- Tietämys tehtävätilanteesta. Tällä tietämyksellä tarkoitetaan käyttäjän käsitystä siitä, mitä kaikkea käyttäjä voi tehdä kyseisessä tilanteessa käytössä olevilla asioilla. Tähän samaan tietämykseen sisältyy myös käyttäjän tietous siitä, miten eri tehtävätilanteet liittyvät toisiinsa.

- Tietämys tarvittavien tehtävien suorituksesta. Tämä tietämys muodostuu GOMS-mallin peruspilareista, eli tavoitteista, menetelmistä, operaattoreista ja valintasäännöistä.

Kieras [1988] kuvasi tekniikkaa NGOMSL-mallin rakentamiseksi siten, että malli koostuu ylhäällä määritellyistä korkean tason tavoitteista, joiden rakennetta tarkennetaan mallissa alaspäin mentäessä. Alaspäin mennään niin kauan, kunnes kyseisellä tasolla on jäljellä vain metodeja, jotka sisältävät primitiiviseksi luokiteltuja operaattoreita. Näitä operaattoreita tyypillisesti ovat KLM-mallin tapaiset konkreettiset tapahtumat. Huolellisesti toteutettuna NGOMSL-mallin avulla voidaan ennustaa tehtävän tavoitteen suoritusaika, operaattorien suoritusjärjestys ja käyttäjältä kuluva aika jonkin menetelmän oppimiseen.

NGOMSL Statements	Executions	External Operator Times
Method for goal: Move text	1	
Step 1. Accomplish goal: Cut text.	1	
Step 2. Accomplish goal: Paste text.	1	
Step 3. Return with goal accomplished.	1	
Method for goal: Cut text	1	
Step 1. Accomplish goal: Highlight text.	1	
Step 2. Retain that the command is CUT, and accomplish goal: Issue a command.	1	
Step 3. Return with goal accomplished.	1	
Method for goal: Paste text	1	
Step 1. Accomplish goal: Position cursor at insertion point.	1	
Step 2. Retain that the command is PASTE, and accomplish goal: Issue a command.	1	
Step 3. Return with goal accomplished.	1	
Selection rule set for goal: Highlight text	1	
If text-is word, then accomplish goal: Highlight word.		
If text-is arbitrary, then accomplish goal: Highlight arbitrary text.	1	
Return with goal accomplished.	1	
Method for goal: Highlight word		
Step 1. Determine position of middle of word.		
Step 2. Move cursor to middle of word.		
Step 3. Double-click mouse button.		
Step 4. Verify that correct text is selected		
Step 5. Return with goal accomplished.		
Method for goal: Highlight arbitrary text	1	
Step 1. Determine position of beginning of text.	1	1.20
Step 2. Move cursor to beginning of text.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Determine position of end of text. (already known)	1	0.00
Step 5. Move cursor to end of text.	1	1.10
Step 6. Shift-click mouse button.	1	0.48
Step 7. Verify that correct text is highlighted.	1	1.20
Step 8. Return with goal accomplished.	1	
Method for goal: Position cursor at insertion point	1	
Step 1. Determine position of insertion point.	1	1.20
Step 2. Move cursor to insertion point.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Verify that correct point is flashing	1	1.20
Step 5. Return with goal accomplished.	1	
Method for goal: Issue a command	1	
Step 1. Recall command name and retrieve from LTM the menu name for it, and retain the menu name.	1	
Step 2. Recall the menu name, and move cursor to it on Menu Bar.	1	1.10
Step 3. Press mouse button down.	1	0.10
Step 4. Recall command name, and move cursor to it.	1	1.10
Step 4. Recall command name, and verify that it is selected.	1	1.20
Step 5. Release mouse button.	1	0.10
Step 6. Forget menu name, forget command name, and return with goal accomplished.	1	

Predicted Procedure Learning Time = 801 sec
Total Predicted Execution Time = 16.38 sec

Kuva 5: Aiemmin mainitun tekstinsiirtoesimerkin esitysmuoto NGOMSL-mallilla [Kieras, 1994, s. 16].

Ylläolevasta kuvasta voidaan havaita ero aiemmin mainittuun CMN-GOMS-malliin, kun tarkastellaan mallin tapaa esittää ohjelmiston valikkojen takaa löytyviä toimintoja, kuten esimerkiksi esimerkissä käytetty “cut”-toiminto. Näitä toimintoja NGOSML-mallissa kuvataan “issue-command”-apumenetelmän avulla. Tämän menettelyn etuna on se, että samankaltaisten valikkotoimintojen lisääminen malliin on helpompaa kuin CMN-GOMS-mallissa. Tämän avulla käyttäjän

muistikuorma voidaan huomioida suunnitteluprosessissa. NGOMSL-mallin ennustamat suoritus- ja oppimisajat ovat tarkkoja ainoastaan niissä tapauksissa, joissa käyttäjä tuntee käytetyt operaattorit. Tämä johtuu siitä, että NGOSML-malli ei sisällä tietoa siitä, miten yksittäinen operaattori suoritetaan tai miten käyttäjä sen oppii. NGOMSL-malli sisältää ainoastaan tietoa siitä, mitä operaattoreita tarvitaan ja missä järjestyksessä.

5.3.4 CPM-GOMS

CPM-GOMS-malli pohjautuu suoraan MHP:hen toisin kuin edellisen kappaleen CMN-GOMS-malli ja siten eroaa ajattelutavaltaan huomattavasti CMN-GOMS- tai KLM-mallista. Eroista huolimatta CPM-GOMS-mallin luominen lähtee CMN-GOMS-mallista [Carroll, 2003, s. 79]. Lyhenteelle CPM on kaksi vaihtoehtoa:

1. Cognitive-Perceptual-Motor. Tämä lyhenne muodostaa CPM-GOMS-mallissa käytettävien operaattoreiden juuret.

2. Critical-Path-Method. Tämä on yksi mallista saatava tulos, josta näkee määritellyn tehtävän suoritusajan.

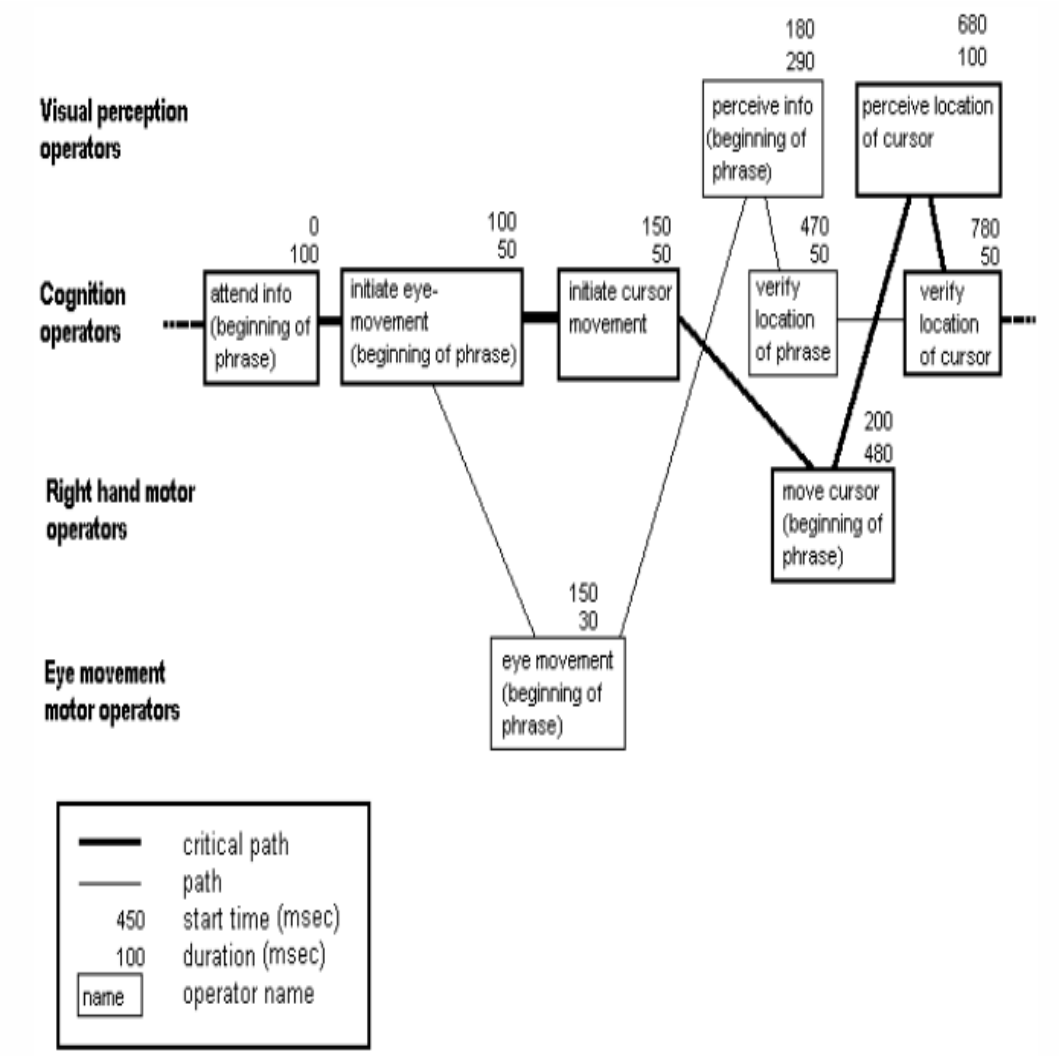
CPM-GOMS-malli esitetään graafisesti projektin hallinnan puolelta tutulla PERT-kaaviolla. CPM-GOMS-mallin kohdalla PERT-kaaviolla voidaan esittää tavoitteen saavuttamiseen tarvittavat operaattorit sellaisessa muodossa, että siitä pystyy päättelemään tavoitteen saavuttamiseen tarvittavan minimiajan summaamalla polun varrella olevien operaattorien suoritusajat. Vahvasta MHP-pohjasta johtuen CPM-GOMS-mallissa informaation käsittelyn oletetaan tapahtuvan eri tietopankkien ja prosessoreiden avulla. Prosessorit CPM-GOMS-mallissa jakautuvat seuraaviin kategorioihin, joista muodostuu myös prosessessointiketju:

1. Havainnointiprosessori, jonka avulla ympäristöstä havaittavissa oleva informaatio talletetaan tietopankkeihin.

2. Tiedostamisprosessori, joka käsittelee tietopankkeihin tullutta informaatiota ja arvioi tarpeellisten jatkotoimien tekoa.

3. Motorinen prosessori, jonka vastuulla on jatkotoimien suorittaminen motorisella tasolla (esimerkiksi näppäimen painallus tai hiiren kursorin liikuttaminen).

MHP-periaatteita mukaillen CPM-GOMS-mallissa näiden prosessoreiden ajatellaan toimivan myös rinnakkain. Esimerkkinä rinnakkaisajosta voi toimia tilanne, jossa havainnointi- ja motorinen prosessori toimivat rinnakkain, kun käyttäjä liikuttaa hiiren kursoria, ennen kuin käyttäjä on paikallistanut näytöltä silmillään liikkeen loppupisteen. Aikaisemmin mainitussa esimerkissä tämä tilanne voi esiintyä, kun käyttäjä ryhtyy siirtämään kursoria lauseen alkuun samalla etsien silmillään lauseen alkukohtaa näytöltä. Mahdollisen rinnakkaisajon huomioiminen on merkittävä ero CPM-GOMS-mallin hyväksi verrattuna KLM- tai CMN-GOMS-malliin. John ja Kieras esittivät tämän prosessoreiden rinnakkaisajoesimerkin tekstinsiirtotapauksesta artikkelissaan seuraavalla tavalla PERT-kaaviolla.



Kuva 6: Esimerkki CPM-GOMS-mallin rinnakkaisajosta [John & Kieras, 1996b].

Ylläolevassa kaaviossa kriittinen polku on merkattu vahventamalla avainasemassa olevien operaattorien laatikoita ja niiden välisiä yhteyksiä. Tätä polkua seu-

raamalla voidaan laskea tavoitteen saavuttamiseen kuluva aika, mikäli mallissa tehdyt oletukset toteutuvat. Tässä esimerkin tapauksessa aikaan vaikuttava oletus on, että oikea käsi voi aloittaa hiiren kursorin siirron ennen kuin silmät ovat havainneet lauseen alkukohdan ja käsitelleet tämän informaation. Ylläoleva esimerkki CPM-GOMS-mallista on hyvin yksinkertainen, eikä se sisällä mitään monimutkaista päätöksentekoprosessia. Laajemmissa käyttökohteissa CPM-GOMS-malli vaatii huolellista ja syvällistä paneutumista prosesseihin hyvän lopputuloksen takaamiseksi. Tästä syystä mallin käytölle on varattava projektissa riittävästi aikaa ja resursseja.

5.4 GOMS-mallin käyttö

GOMS-mallin avulla sovelluksia voidaan tutkia kvantitatiivisesti ja kvalitatiivisesti. Nämä ominaisuudet tekevät GOMS-mallista sopivan työkalun analysoimaan erityisesti systeemejä, joissa käyttäjän oletetaan haluavan tehokkaampia työskentelytapoja oppimiskäyrän alkuvaiheen jälkeen. Mallin käyttö on hyödyllistä jo suunnitteluvaiheessa, jolloin erilaisten toteutustapojen vaikutusta voidaan arvioida ennen itse toteutusta.

5.4.1 Valinta eri GOMS-mallien välillä

Ennen käytettävän GOMS-mallin valintaa on hyvä analysoida käyttäjiltä vaadittavat tehtävät siten, että mallin käyttö on mahdollista kriittisten ominaisuuksien puolesta. Nämä tehtävän kolme kriittistä ominaisuutta GOMS-mallin valinnan kannalta ovat seuraavat [John & Kieras, 1994, s. 28]:

- Rutiininomaisen taidon merkitys tehtävän suorituksessa.
- Peräkkäismuotoisen suoritustavan olemassaolo.
- Aika, joka vuorovaikutustilanteessa kuluu tietokoneen tai jonkin muun ulkopuolisen operaattorin kontrollissa käyttäjän sijaan.

Taito ominaisuutena tehtävän suorituksessa saattaa vaihdella äärimmäisestä ongelmanratkaisusta, jossa käyttäjä ei tiedä, miten tehtävä suoritetaan ja joutuu etsimään ratkaisun rutiininomaiseen kognitiiviseen taitoon, jonka avulla käyttäjä tietää, miten tehtävä tulee suorittaa ilman varsinaista ratkaisun etsimistä. Tämän tarkka empiirinen mittaaminen ei ole yksinkertaista ja GOMS-mallit eivät tarjoa siihen valmista työkalua.

Tehtävien peräkkäisyys tai rinnakkain suoritettavuus. Osa tehtävistä voidaan luokitella käytännössä peräkkäinsuoritettaviksi, kuten esimerkiksi tekstinmuokkaus. Joissakin tehtävissä puolestaan esiintyy päällekkäisiä ja rinnakkain suoritettavia toimintoja, jolloin niitä ei voi pitää peräkkäismuotoisina suoritusmuodoltaan. Tämän lisäksi käyttäjien kokemuspohja vaikuttaa tehtävien suoritusmuotoon. Aikaisemmin mainitussa "tekstin siirto"-tehtävän kohdalla voidaan olettaa kokemattoman käyttäjän etenevän tehtävän läpi askel askeleelta, kun kokeneempi käyttäjä voi ryhtyä siirtämään kursoria jo ennen kohteen näkemistä.

Kontrollin sijainti. Tietokoneen kannalta tehtävät voidaan jakaa karkeasti kahteen osaan: passiivisiin ja aktiivisiin. Passiivisissa tehtävissä tietokone pääsääntöisesti odottaa syötettä käyttäjältä, ja tällöin tavoitteen saavuttamisen kannalta käyttäjällä on kontrolli etenemisnopeuteen ja tehtävien ajoitukseen. Aikaisemmin mainittu "tekstin siirto"-tehtävä on hyvä esimerkki koneen kannalta passiivisesta tehtävästä. Aktiivisissa tehtävissä tietokone puolestaan kykenee toimimaan itsenäisesti, ja tällöin kokonaissuoritusaikaan vaikuttaa enemmän käyttäjän kyky reagoida koneen antamaan informaatioon.

Näiden ylläolevien seikkojen jälkeen voidaan käyttää alla olevaa Johnin ja Kieraksen esittelemää kaaviota [John & Kieras, 1994, s. 29] käytettävän GOMS-mallin valintaan.

Task Type / Information Type	Routine Cognitive Skill			
	Passive System		Active System	
	Sequential	Parallel	Sequential	Parallel
Functionality: Coverage	Any GOMS	Any GOMS	Any GOMS	Any GOMS
Functionality: Consistency	NGOMSL		NGOMSL	
Operator Sequence	CMN-GOMS NGOMSL	CPM-GOMS	see text	CPM-GOMS
Execution Time	KLM CMN-GOMS NGOMSL	CPM-GOMS	see text	CPM-GOMS
Procedure Learning Time	NGOMSL		NGOMSL	
Error Recovery Support	Any GOMS	Any GOMS	Any GOMS	Any GOMS

Kuva 7: Käytettävissä olevat GOMS-mallit eri kombinaatioille [John & Kieras, 1994, s. 29].

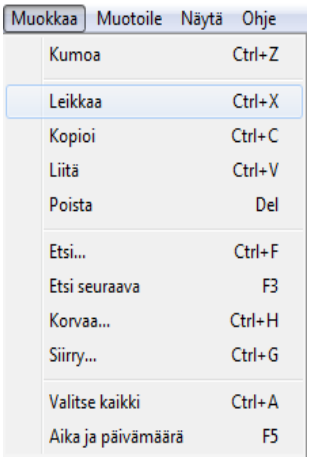
Mallin valintaan vaikuttaa myös halutun lopputuloksen päättäminen. Esimerkiksi CMN-GOMS- ja NGOMSL-mallien ollessa ohjelmamuotoisia ne on kuvattu eksplisiittisesti. Tällöin niiden avulla on mahdollista ennustaa käyttäjän käyttämät menetelmät ja niihin vaadittujen operaattorien suoritusjärjestys. Samalla malli itsessään toimii ohjelmistokehittäjälle ohjelmointitehtävässä tukirankana. Mikäli on tiedossa, että analysoitava kohde on hyvin suoraviivainen ja lopputuloksesta kiinnostaa vain suoritusaika, niin silloin valinta saattaa kääntyä KLM-mallin suuntaan, koska se kykenee tuottamaan halutun tuloksen helpommin hivenen kevyemmän rakenteensa vuoksi.

5.4.2 Kvantitatiivinen tutkimus

GOMS-malli sopii hyvin kvantitatiiviseen tutkimukseen sovelluksien kohdalla, koska sen avulla on mahdollista saada hyviä arvioita absoluuttisesta suoritusajasta ja suhteellisesta oppimisajasta. Mallin avulla sovelluksella eri tavoitteiden saavuttaminen voidaan analysoida yksityiskohtaisesti ja jakaa prosessi osiin, jolloin vaihtoehtoisten metodien käyttöä saman alitavoitteen saavuttamiseen voidaan vertailla keskenään.

5.4.3 Kvalitatiivinen tutkimus

GOMS-malli sisältää luonnostaan prosessin, joka on hyvin tarkka määrittelemään tavoitteen tai maalin saavuttamiseen tarvittavaa käyttäjän omaamaa tietoa. Tämän vuoksi malli tuottaa tehtäväorientoitunutta dokumentaatiota, jota voidaan käyttää koulutusohjelmien, avustussysteemien ja systeemin itsensä kehittämiseen [Carroll, 2003, s. 62]. Aikaisemmin mainitun tekstin leikkaus- ja liimausesimerkin yhteydessä GOMS-mallin tuottamaa tietoa voitaisiin käyttää käytössä olevan tekstieditorin avustussysteemin laadun tutkimiseen. Tässä esimerkissä näkyisi käyttäjän opastus siten, antaako editori esimerkiksi vinkkejä tottumattomalle käyttäjälle näppäinyhdistelmien olemassaolosta, kuten alla olevassa kuvassa.



Kuva 8: Notepadin muokkaa-valikko.

5.4.4 Dokumentaation parantaminen

GOMS-mallin avulla voidaan myös parantaa ohjelmiston tai järjestelmän dokumentaation laatua vertaamalla sitä mallin tuottamaan tehtäväorientoituneeseen dokumenttiin. Tällä tavoin mahdolliset aukot eri toimintojen käyttöohjeissa voidaan havaita.

Dokumentaatiota voi parantaa myös siten, että se tehostaa käyttäjän oppimisprosessia. Tämän voi saavuttaa esimerkiksi GOMS-mallilla, jonka analyysin tuloksia voi käyttää dokumentaation parantamiseen. Työkaluina toimivien ohjelmistojen kohdalla tämä voisi tarkoittaa käyttöohjeissa esimerkkikäyttötapausten purkamista yksityiskohtaisiksi askeleiksi ja niiden suorittamiseen vaadittavan tiedon vertaamista dokumentaatiosta löytyvään tietoon.

Richard Gong ja Jay Elkerton [1990] tekivät tutkimuksen GOMS-mallilla parannellun käyttöohjeen vaikutuksesta realistisessa käyttötapauksessa. Tutkimuksen kokeessa he jakoivat koeryhmän jäsenet neljään eri ryhmään ja jokaisella ryhmällä oli samasta ohjelmistosta erilainen käyttöohje käytössä. Koehenkilöille annetut käyttöohjeet olivat seuraavat:

1. Alkuperäinen käyttöohje oli ohjelmiston mukana tuleva 69-sivuinen käyttöopas ilman mitään muokkauksia. Luonteeltaan alkuperäinen käyttöohje oli systeemiorientoitunut ja pyrki antamaan käyttäjälle etusijassa käsityksen koko ohjelmistosta ennen tiettyjen tehtävien esimerkkisuorituksien kuvaamista. Tutkijoiden mukaan noin kolmannes tästä käyttöohjeesta oli ennakkotietoa, jota käyttäjän oletettiin hallitsevan ennen varsinaisen tehtävän suorittamiseen vaadittujen metodien opiskelua.

2. Lyhennetty käyttöohje alkuperäisestä, jonka tarkoituksena oli selvittää yksinkertaisesti, onko asioiden tiiviimmällä esitysmuodolla vaikutusta oppimistapahtumaan ja laatuun. Tiivistetyn käyttöohjeen pituus oli 24 sivua.

3. Proseduraalinen käyttöohje, joka oli laadittu GOMS-mallin avulla. Pituutta tällä käyttöohjeella oli 21 sivua. Päätarkoituksena tällä versiolla oli nopeasti orientoida käyttäjä tunnistamaan ohjelmiston päätason tavoitteet ja nämä samat päätason tavoitteet toimivat käyttöohjeen kappaleina. Alkuperäiseen täysmittaiseen käyttöohjeeseen verrattuna tämä proseduraalinen käyttöohje sisälsi hyvin vähän orientaatiota systeemiin itseensä.

4. Virhetilanteesta palautumisen käyttöohje sisälsi samat tekstit kuin proseduraalinen käyttöohje. Tämän lisäksi tähän käyttöohjeversioon oli sisällytetty ohjeistusta mahdollisten virhetilanteiden tunnistamisesta, välttämisestä ja niistä toipumiseen. Nämä lisätiedot toivat pituutta kolme sivua ja täten kokonaispituus oli 24 sivua. Tämä lisätieto oli jalostettu ohjelmiston perehdytystyöpajoilta, josta aloittelevien käyttäjien virhetilanteet oli taltioitu.

Tämä lisätieto oli sijoitettu proseduraalisen käyttöohjeen päälle siten, että virhetilanteen välttämisen kohdalla tieto oli sijoitettuna ennen proseduurin runkotekstiä ja virhetilanteesta toipumisen kohdalla runkotekstin jälkeen.

Yllämainituista käyttöohjeista voidaan muodostaa kaksi paria, joista ensimmäinen on alkuperäinen ja lyhennetty käyttöohje. Toisena parina proseduraalinen ja virhetilanteesta palautumisen käyttöohje. Ero näiden parien välillä on selkeä ja sen voi helposti todeta vilkaisemalla sisällysluetteloa.

TABLE OF CONTENTS

Kuva 9: Alkuperäisen käyttöohjeen sisällysluettelo [The University of Michigan, 1988].

Kuten ylempänä mainittiin, niin kappaleet ja sisällysluettelo on laadittu systeemiorientoidusti esittämällä jokainen käyttöliittymän komento omassa kohdas-

saan. Tämä näkyy selkeästi kappaleiden jaottelussa ja nimeämisessä, jotka palvelevat hyvin koko ohjelmiston esittelyssä. Käyttäjän toiminannan tehokkuuden kannalta tällainen ei kuitenkaan ole optimaalinen lähestymistymistapa. Esimerkkinä voidaan mainita vaikka ohjelmiston käynnistämisen ohjeiden löytyminen vasta sivulta 20, jota ennen on pelkkää perehdytysmateriaalia. Vastaavalla logiikalla tehtynä monimutkaisemman ohjelmiston kohdalla itse ohjelmiston käynnistämisen ohjeistus voisi löytyä vasta parinsadan perehdytyssivun jälkeen. Käytännössä tällainen ei olisi tehokasta varsinkaan sellaisten ohjelmistojen kohdalla, joissa käyttäjien ei oleteta käyttävän ohjelmiston jokaista komponenttia.

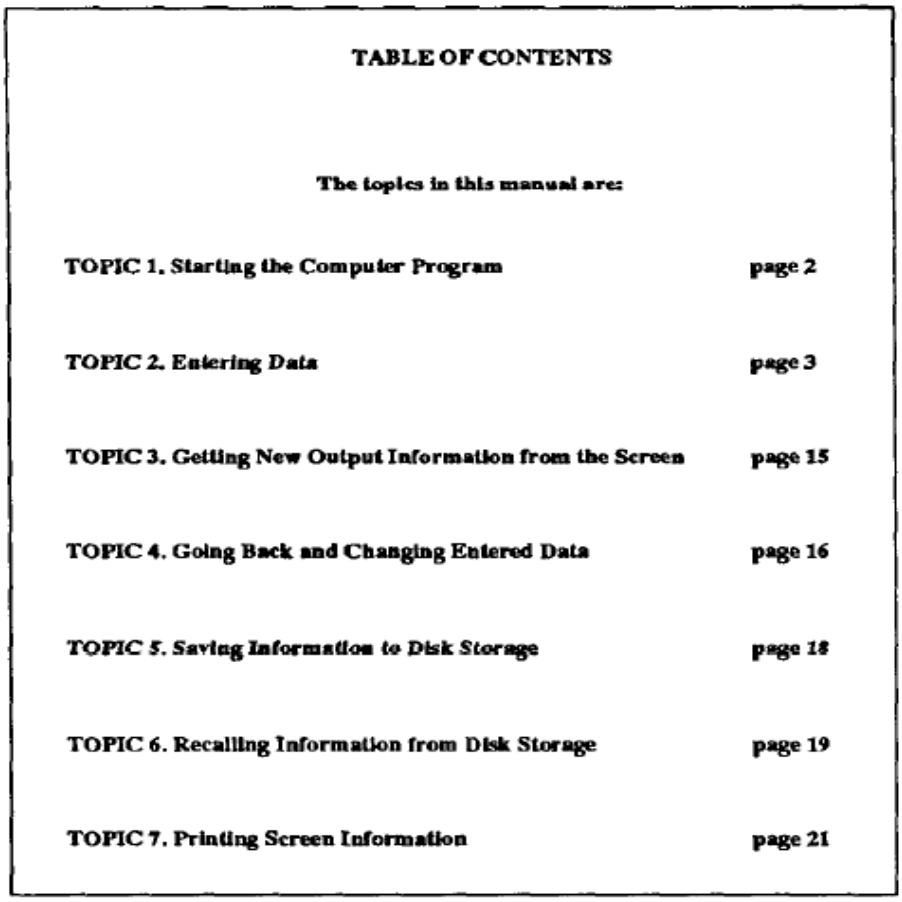


TABLE OF CONTENTS

The topics in this manual are:

Kuva 10: Proseduraalisen käyttöohjeen sisällysluettelo [The University of Michigan, 1988].

On helppo huomata pelkästään sisällysluettelojen tekstimäärää vertaamalla, että sisällysluettelot on toteutettu eri periaatteella. Sisällysluettelossa mainittuja päätasojen kappaleiden nimiä vertaamalla havaitaan hyvin käyttöohjeiden laadintaperiaatteellinen ero. Proseduraalisen käyttöohjeen sisällysluettelo (Kuva 10)

on laadittu GOMS-mallin mukaisten päätason tavoitteiden mukaan. Kappaleiden tekstuaalinen asu on hiottu niin, että niistä uusikin käyttäjä kykenee ymmärtämään hakemaansa toiminnollisuutta vastaavan kappaleen. Tälläinen käyttöohje on yksi tapa auttaa uutta käyttäjää saavuttamaan tavoitteensa nopeammin.

Itse koetilanteessa kokeeseen osallistuneille henkilöille annettiin ennalta määritelty tehtävälista ohjelmistolla suoritettavaksi. Tehtävälista oli laadittu siten, että se vastaisi realistista ongelmanratkaisua ohjelmistolla. Tehtävälistan laatijat eivät olleet tutustuneet koejärjestelyä varten laadittuihin eri käyttöoppaan versioihin ja näin varmistettiin, ettei yhtä käyttöopasversiota suosittaisi tehtävän suunnittelussa. Itse tehtävät pystyttiin luokittelemaan kolmeen seuraavaan ryhmään:

1. Uusi tehtävä, jonka suoritus vaatii uusien proseduurien opiskelua.

2. Identtinen tehtävä, jonka suorittamiseen vaaditaan aikaisemmin opittujen metodien tai proseduurien käyttämistä eri syöte- ja tulosteparametreillä.

3. Siirtymätehtävä, joka vaatii aikaisemmin opittujen proseduurien ketjuttamista peräkkäisten analyysien tekoa varten.

Koetilanteessa kerättiin tietoa seuraavista asioista koehenkilöiden suorituksesta käyttämällä ajanottokelloa ja aikaleimattua videomateriaalia tehtävän suorituksesta:

- Yhden tehtävän suorittamiseen kulunut aika.

- Ohjekirjaan katsomiseen kulunut aika tehtävän aikana.

- Tehtyjen virheiden määrä tehtävän suorituksen aikana.

Näiden lisäksi käyttäjiltä myös pyydettiin arviointia käyttöohjeesta heidän aikaisemmin käyttämiinsä käyttöohjeisiinsa verrattuna.

Tutkimuksen kokonaistuloksia tarkasteltaessa GOMS-mallin käytön positiiviselle vaikutukselle löytyy merkittävä etua, kun tarkastellaan tehtäväkokonaisuuteen kulutettua aikaa ja tehtyjen virheiden määrää. *Post Hoc* -analyyseissä Fisherin LSD-metodilla tutkimuksessa havaittiin olevan merkitsevä eroavaisuus alkuperäisen käyttöohjeen ja kummankin proseduraalisen käyttöohjeen välillä. Virhetilanteesta toipumiseen suunnitellun käyttöohjeen kanssa saatiin suoritusaikaan

41 % parannus alkuperäiseen käyttöohjeeseen verrattuna. Sama metodi toisen proseduraalisen ohjeen kanssa tuotti 25 % eron alkuperäiseen käyttöohjeeseen nähden. Tutkimuksessa ei paljastunut samassa yhteydessä merkitsevää eroavaisuutta, kun samaa tyyliä edustavia kahta käyttöohjetta verrattiin keskenään.

Tehtävissä tehtyjä virheitä tarkasteltaessa tutkimuksessa havaittiin, että merkitsevää eroa alkuperäisen käyttöohjeen ja muiden käyttöohjeversioiden välillä löytyy. Samaa merkitsevää eroa esiintyy, kun verrataan virhetilanteesta toipumiseen suunniteltua proseduraalista käyttöohjetta jompaankumpaan alkuperäisistä käyttöohjeista. Merkitsevää eroa ei virheiden määrään löytynyt, kun proseduraalista käyttöohjetta verrattiin alkuperäisiin käyttöohjeversioihin tai virhetilanteesta toipumiseen suunnitellusta proseduraaliseen käyttöohjeeseen (Taulukko 5).

Käyttöohje	Kokonaisaika (min)	Virheiden määrä (kpl)
Alkuperäinen	48,8	17,9
Lyhennetty alkuperäinen	43,5	11,1
Proseduraalinen	35,9	7,7
Virhetilanteesta palautuminen	28,6	5,3
Fisher LSD	8,9	4,1

Taulukko 5: Keskiarvoistetut suoritusajat ja tehdyt virheet koko kokeelle.

Kokonaistuloksissa esiintyneet erot ovat selkeimmin nähtävissä uusien tehtävien teosta koostetuissa tuloksissa. Näiden tuloksien perusteella voidaan havaita, että kokonaisaikojen vertailussa esiintyvät erot selittyvät hyvin pitkälle uusien tehtävien tekoon kulutetun ajan ja tehtyjen virheiden avulla. Lähemmissä analyyseissä selvisi, että vertailuryhmien jäsenet, joilla oli käytössään proseduraalinen käyttöohje, suorittivat tehtävät merkittävästi lyhyemmässä ajassa. Analyysin laskelmien mukaan tämä tarkoitti vähintäänkin 28 % etua alkuperäisten käyttöohjeiden uuden tehtävän suoritukseen nähden. Uusien tehtävien kohdalla kahdella alkuperäisellä käyttöohjeella ei ollut merkitsevää eroa. Sama pätee myös proseduraalisten käyttöohjeiden välisiin tuloksiin. Tehtyjen virheiden kohdalla tulokset eivät olleet käytetyn ajan kanssa samankaltaiset. Tuloksien perusteella heikoimmin tehtävän suorittamisesta suoriutuivat lyhentämättömän alkuperäisen käyttöohjeen saaneet koehenkilöt ja he tekivät vähintään 37 % enemmän virheitä kuin kolmea muuta käyttöohjetyyppiä käyttäneet koehenkilöt. Merkittävästi vähiten virheitä tehtävän suorittamisen yhteydessä tekivät koehenkilöt, joilla oli käytössään virhetilanteista selviytymiseen suunniteltu proseduraalinen käyttöohje. Näiden kolmen muun käyttöohjetyypin käyttäjien tekemien virheiden määrässä uusissa

tehtävissä ei muuten ollut merkitsevää eroa. Tutkimuksen tekijät hakivat selitystä havaitsemilleen suoritusaikaeroille tarkastelemalla koetilanteesta nauhoitettua videota. Videon avulla tutkijat pystyivät erottelemaan suorituksesta käyttöohjeen katsomiseen kuluneen ajan muuhun toimintaan kuluneesta ajasta, joka koos-tui pääsääntöisesti näytön tai näppäimistön katsomisesta. Itse tehtävälistan katsomiseen kulunutta aikaa voidaan pitää videon perusteella merkityksettömänä. Videoanalyysin perusteella tutkimuksessa todettiin,että proseduraalisten käyttöohjeiden käyttäjät kuluttivat 50 % vähemmän aikaa käyttöohjeen katsomiseen kuin henkilöt, joilla oli käytössään jompikumpi versio alkuperäisestä käyttöohjeesta. Itse tehtävän suorittamiseen proseduraalisten käyttöohjeiden käyttäjillä kului vähintään 30 % vähemmän aikaa kuin alkuperäisten käyttöohjeiden käyttäjillä.

Käyttöohje	Kokonais-aika uudelle tehtävälle	Virheet	Käyttöohjeen lukuaika uudelle tehtävälle	Uuden tehtävän suoritu-saika
Alkuperäinen	34,6	16,2	12,3	22,3
Lyhennetty alkuperäi-nen	31,1	9,8	10,8	20,3
Proseduraalinen	22,4	6,6	6,7	15,7
Virhetilanteesta palau-tuminen	19,3	3,5	6,2	13,1
Fisher's LSD (0,05)	6,9	3,7	3,8	4,3

Taulukko 6: Keskimääräiset ajat ja virheet uusissa tehtävissä.

Identtisten ja peräkkäisten siirtymätehtävien keskimääräisissä suoritusajoissa ei havaittu merkitseviä eroja käytössä olleiden käyttöohjeiden välillä. Virhetilanteista palautumiseen suunnitellun proseduraalisen käyttöohjeen kohdalla voitiin havaita vähäistä eroa siirtymätehtävien kohdalla muihin käyttöohjeversioihin nähden ja tämä antaa mahdollisuuden spekulaatiolle siitä, auttoiko juuri käyttöohje suoritusajan parantamiseen. Nämä tulokset olivat sinänsä odotettavia, koska käyttäjällä voitiin olettaa olevan tehtävän suorittamiseen vaadittavat tiedot tuoreessa muistissa edellisten tehtävien jälkeen.

Käyttöohje	Identtinen tehtävä (min)	Siirtymätehtävä (min)
Alkuperäinen	3,2	11,0
Lyhennetty alkuperäinen	2,5	9,8
Proseduraalinen	3,3	10,3
Virhetilanteesta palautuminen	2,4	6,9

Taulukko 7: Keskimääräiset ajat identtisille ja peräkkäisille tehtäville.

Varsinaisten tehtävien teon jälkeen koehenkilöiltä kysyttiin tutkimustilanteen lopuksi arviota käyttämästään käyttöohjeesta. Arvosteluasteikko oli välillä 1-5, jossa 1 merkitsi "Käyttöohjeesta ei ollut juurikaan hyötyä" ja 5 puolestaan "Käyttöohje oli todella hyödyllinen". Käyttöohjeen rakenteen hyödyllisyyttä tarkasteltaessa havaittiin merkitsevää eroa alkuperäisen käyttöohjeen ja kolmen muun käyttöohjetyypin välillä. Merkitsevää eroa havaittiin myös käyttöohjeen pituuden arvostelussa alkuperäisen käyttöohjeen ja muiden käyttöohjetyyppien välillä. Arvostelussa oli yllättävänä seikkana se, että käyttöohjetyypin hyödyllisyydestä virhetilanteesta toipumisen kohdalla ei löytynyt merkitsevää eroa, vaikka tehtävien suorituksessa oli havaittavissa selkeää etua virhetilanteista palautumiseen suunnitellun proseduraalisen käyttöohjeen hyväksi.

Käyttöohje	Käyttöohjeen rakenne	Käyttöohjeen pituus	Apu virhetilanteesta toipumiseen
Alkuperäinen	3,5	3,1	3,0
Lyhennetty alkuperäinen	3,8	3,9	3,1
Proseduraalinen	4,4	4,1	2,9
Virhetilanteesta palautuminen	4,2	3,9	3,5
Fisher's LSD	0,6	0,7	NA

Taulukko 8: Subjektiiviset arvioinnit ohjekirjatyypin hyödyllisyydestä.

Tutkimuksen tuloksien perusteella GOMS-metodin käytölle on paikkansa käyttöohjeiden suunnittelussa.

5.4.5 Yhteenveto GOMS-mallista

GOMS-malli on perusteiltaan nykyisin jo yli 30 vuotta vanha, ja sitä voidaan pitää yhtenä varteenotettavana työkaluna käyttöliittymää tai ohjeita suunniteltaessa. Mallin hyvänä puolena voidaan pitää työmäärän kohtuullisuutta, kun sen avulla arvioidaan tehtävien vuorovaikutustapahtumia. Hyväksi ominaisuudeksi voidaan myös lukea mallin kyky ennustaa käyttöliittymään suunniteltujen muutosten vaikutukset ennen varsinaista toteutusta ja testausta.

GOMS-mallin selkeänä heikkona puolena on sen oletus siitä, että käyttäjä tietää aina mitä on tekemässä. Tästä syystä mallin avulla on vaikea ennustaa aloittelijoiden vuorovaikutusta suunnitellun järjestelmän kanssa. Lisäksi malli arvioi puutteellisesti käyttäjän kognitiivisten toimintojen ja ympäristön vaikutusta tavoitteen saavuttamisessa.

6 Tapaustutkimus OptoFidelityn käyttöliittymistä

Tapaustutkimuksessa on tarkoituksena käydä läpi keskeisimpiä käyttötapoja OptoFidelityn työkaluilla ja keskittyä kuvaamaan niihin liittyviä käytettävyysongelmia. Käytettävyysongelmia tarkastellaan niin käytön tehokkuuden kuin tahattomiin virhetilanteisiin päätymisen kannalta. OptoFidelityn asiakkailla voi olla omia käyttötapauksia eri tarkoituksineen työkaluille ja näiden kaikkien kartoitus olisi hyödyllistä, mutta käytettävien resurssien vuoksi mahdotonta.

OptoFidelityn työkalut sopivat tämän tutkielman aiheeksi hyvin, koska ne edustavat käyttöliittymältään tyypillisiä insinöörityökaluja. Insinöörityökaluilla tämän tutkielman puitteissa keskitytään ohjelmistoihin, jotka eivät päädy yleensä tavalliselle kuluttajalle asti, vaan niitä käytetään pääsääntöisesti yritysten sisällä tuotekehitystehtävissä. Tämä tarkoittaa yleensä sitä, että käyttöliittymät eivät ole loppuun asti viimeisteltyjä. Useimmissa tapauksissa tämä puolestaan tarkoittaa sitä, että käyttöliittymissä esiintyvät käsitteet ja logiikat voivat olla keskivertokuluttajalle tuntemattomia. Ylläolevan kuvauksen mukaisia insinöörityökaluja voidaan karkeasti jaotella kahteen tyyppiin:

- Yrityksen sisäiseen käyttöön luodut työkalut. Näitä kutsutaan yleisesti myös niin sanotuiksi talon sisäisiksi työkaluiksi. Nämä työkalut pysyvät yrityksen sisällä seuraavista syistä:
  - Yrityksen henkisen pääoman suojelu. Tämä tarkoittaa sitä, että työkalu sisältää tietoutta tai jotain muuta arkaluonteista tietoa, jota ei haluta luovuttaa yrityksen ulkopuolelle.
  - Työkalun hyvin tarkka työskentely-ympäristö, mistä johtuen työkalun käyttö yrityksen ulkopuolella olisi käytännössä hyvin vaikeaa tai täysin mahdotonta. Tästä johtuen yrityksellä ei ole taloudellista perustetta yrittää myydä tai antaa työkalua ulkopuolisille.
  - Resurssien riittämättömyys työkalun viimeistelyyn sille asteelle, jota edellytetään ulkopuolisille luovutettaville työkaluille. Tästä johtuen työkalu pidetään yrityksen sisällä.
- Muille yrityksille tai tahoille myytävät työkalut. Näitä työkaluja viimeistellään myynnin edistämiseksi enemmän kuin yrityksen sisäiseen käyttöön

tarkoitettuja. Viimeistely B2B-ympäristössä myytävissä työkaluissa oletettavasti tapahtuu pääasiassa toiminallisuuden ja regressiotestauksen puolella. Käyttöliittymän ulkoasun viimeistelyn merkitys vaihtelee hyvin paljon eri alojen välillä. Graafisten tai taiteellisten työkalujen kohdalla käyttöliittymän ulkoasuun panostetaan todennäköisesti enemmän kuin muilla aloilla, joissa käyttöliittymän ulkoasu ei näyttele kovinkaan suurta roolia. OptoFidelityn kehittämät työkalut, jotka tässä tutkielmassa mainitaan, kuuluvat tähän viimeksi mainittuun ryhmään.

Käyttöliittymän tai käyttökokemuksen viimeistelemättömyys on yleisempää niin sanotuilla talonsisäisillä työkaluilla. Pääsyynä tähän on se, että aikaa käyttöliittymän tai käyttökokemuksen viimeistelyyn ei ole, vaikka niiden huono laatu laskisikin työkalun kokonaisvaltaista tehokkuutta. Työkalun kehityksen alkupäässä näihin asioihin olisi helppo vaikuttaa, mutta se voi jäädä tekemättä monestakin eri syystä. Yhtenä syynä voisi pitää sitä, että työkalua suunniteltaessa sen elinkaaren oletettiin olevan lyhyt, josta luonnollisesti muodostuu hyvä syy sille, ettei käyttöliittymään tai käyttökokemukseen ole järkevää kuluttaa ylimääräistä aikaa. Tämä ongelma korostuu tapauksissa, joissa tällaisesta suunnitteluvaiheessa lyhyen elinkaaren työkalusta muodostuu pitkäkestoinen käytäntö ja työkalua koetetaan laajentaa. Tämän voidaan olettaa hyvin harvoin tuovan parannuksia käyttöliittymään tai varsinaiseen käyttökokemukseen, koska lisätoimintojen voi yleisesti olettaa tuovan käyttöliittymään lisää kontrolleja tai indikaattoreita ja nämä voivat sekoittaa käyttöliittymää entisestään. Jotkin näistä hiomattomista työkaluista voivat olla useita vuosia käytössä ja niistä muodostuu käytäntö. Tästä puolestaan seuraa, että huonoon käyttöliittymään tai käyttökokemukseen yksinkertaisesti totutaan ja sitä opitaan sietämään. Pahimmillaan tämä vaikuttaa uusienkin työkalujen suunnitteluun, koska tuttujen ratkaisujen tuominen uuteenkin työhön on luontaista henkilöille.

Saponas, Prabaker, Abowd ja Landay [2006] tutkivat aihetta ja löysivät näyttöä sille, että suunnittelijoiden työhön voidaan vaikuttaa ennen suunnittelutyötä esitetyillä materiaaleilla. Itse koejärjestelyssä he hankkivat 44 ammattimaista käyttöliittymäsuunnittelijaa vähintään kahden vuoden alakohtaisella ammattikokemuksella koehenkilöiksi ja jakoivat ryhmän kahden hengen työpareihin. Tämän jälkeen puolelle pareista oli pääsy suunnittelutyön yhteydessä 48 ennakkoon suunniteltuun ratkaisuun, joita voisi hyödyntää heidän tehtävissään. Ennen koetilannetta suunnitellut ratkaisut olivat koejärjestäjien tekemiä. Näihin ratkaisuihin

niiden suunnittelijat hyödynsivät senhetkisiä akateemisia tutkimuksia ja töitä digitaalisen kodin saralta. Jokaiseen ratkaisuun oli kirjoitettu lyhyt kuvaus ongelmasta ja ratkaisusta, jonka kyseinen suunnitelma toteuttaisi. Tehtävänä ryhmillä oli suunnitella digitaaliseen kotiin tuleva järjestelmä, joka viestittäisi käyttäjälleen hänen taloutensa ruokavarastojen inventaarion. Tämän tiedon avulla käyttäjällä olisi muun muassa tieto siitä, mitä ruokia olisi käytettävissä ja mitkä ovat lopussa. Saman tiedon pohjalta järjestelmä voisi antaa palautetta käyttäjälleen ruokakulutuksen perusteella ja myös ehdottaa uusia reseptejä tämän saman tiedon pohjalta. Aikaa tehtävän suoritukseen ryhmillä oli tasan kaksi tuntia. Tehtävien teon jälkeen koejärjestäjät valmistelivat työt arvostelua varten siten, että poistivat töistä kaikki viitteet puolelle ryhmistä annettuihin ennakolta suunniteltuihin ratkaisuihin. Arvostelijoiksi koejärjestäjät valitsivat kolme kokenutta arvostelijaa, joilla oli kokemusta heuristisista arvostelumetodeista. Arvostelijat arvostelivat työt heuristisen arvoinnin lisäksi myös subjek-tiivisesti yksityiskohtaisuuden, täydellisyyden ja laadun suhteen. Arvioiden lisäksi koejärjestäjillä oli käytössään ryhmien suunnittelutyöskentelystä videotallenne, jota muun muassa käytettiin toiselle puolelle ryhmistä annettujen suunnitelmien käytön seuraamiseksi. Koehenkilöitä, joilla oli käytössään koejärjestäjien antamat suunnitelmat, pyydettiin myös antamaan palautetta koetilanteen jälkeen. Palautteessa heiltä kysyttiin mielipiteitä käytössä olleiden suunnitelmien hyödyllisyydestä heidän töissään. Tutkimuksessa havaittiin seuraavia seikkoja:

- Ryhmät, joilla oli käytössä koejärjestäjien ennakkoon tehdyt ratkaisusuunnitelmat, tekivät vähemmän heuristisia virheitä töissään.

- Valmiiden suunnitelmaratkaisujen käyttö on aktiivisinta suunnittelutyön alkuvaiheessa. Tämä päti varsinkin niissä tapauksissa, joissa suunnittelijoilla ei ollut suoraa kokemusta juuri koetilanteen tehtävän ympäristöstä.

- Valmiiden suunnitelmaratkaisujen käytöllä todettiin olevan positiivinen vaikutus uusien ideoiden syntymiseen.

- Valmiita suunnitelmaratkaisujen nimiä ja luokitteluja ryhdyttiin käyttämään ryhmänsisäisissä keskusteluissa suunnittelumallien kanssa.

Saman suuntaisia tuloksia on saatu myös käytännön sosiaalitieteiden puolelta tutkimuksessa [Arad, 2012], jossa Arad järjesti käytännön demonstraation aikaisempien tehtyjen päätöksien vaikutuksesta tuleviin päätöksiin. Käytännön kokeillaan Arad sai tukea niille väitteille, joiden mukaan aikaisemmilla päätöksillä

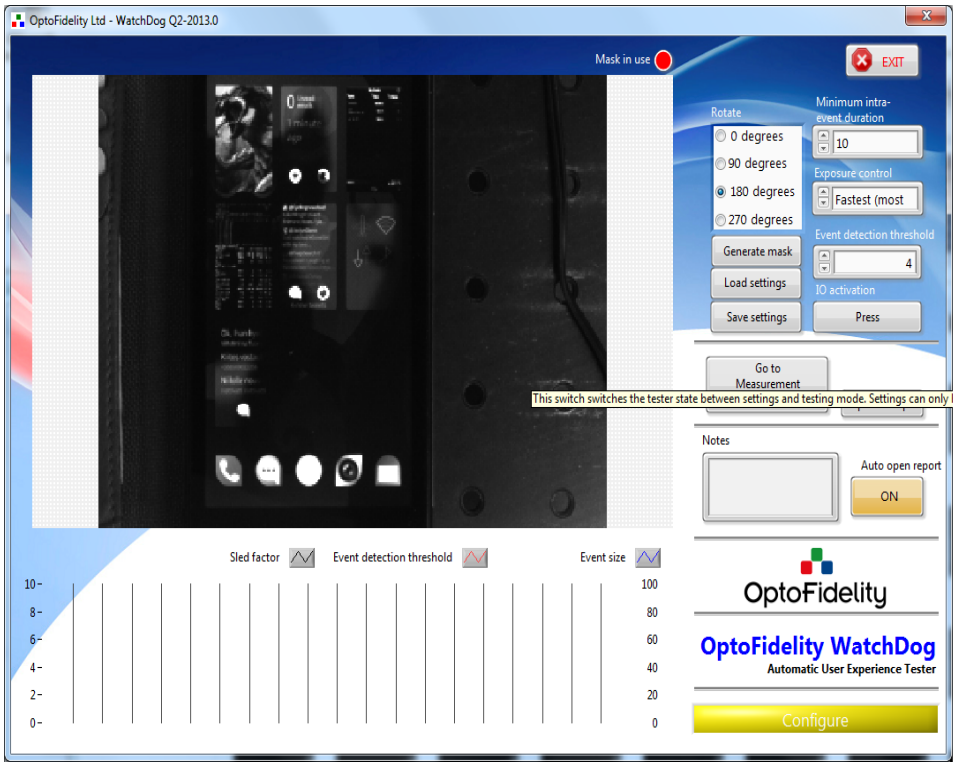
on vaikutusta tuleviin päätöksiin. Saman sosiaalisen mekaniikan voidaan olettaa toimivan myös ohjelmistojen kohdalla suunnittelussa, kun tehdään päätöksiä kehityssuunnista. Kehityssuuntiin vaikuttaa myös toinen sosiaalinen aspekti. Ohjelmistosuunnittelijan ja -kehittäjän ympärillä vallitseva sosiaalinen paine vaikuttaa myös lopputulokseen. Esimerkiksi Tourangeau, Smith ja Rasinski [1997] järjestivät psykologian tutkimuksessaan koetilanteen, jossa vertailtiin sosiaalisen paineen vaikutusta vastausten laatuun eri tilanteissa. Tutkimuksessa havaittiin, että vastausten laatuun ja luotettavuuteen vaikutti sosiaalisen hyväksynnän haku. Gardete puolestaan omassa tutkimuksessaan [Gardete, 2014] sosiaalisesta paineesta havaitsi lentokoneiden lennonaikaisten ostoksien teosta kerätystä tiedosta, että henkilön lähellä olevien muiden henkilöiden toimilla on vaikutusta henkilön omaan käyttäytymiseen. Isoissa yrityksissä tämä ohjelmistojen kohdalla voidaan suhteuttaa esimerkiksi pakonomaisen jatkokehityksen käynnistämiseen ainoastaan siitä syystä, että muun muassa rinnakkaisessa projektissa tehdään niin. Tämä puolestaan saattaa taas puolestaan johtaa siihen, että kyseiseen ohjelmistoon tulee tarpeettomia toimintoja, jotka aiheuttavat tarpeetonta monimutkaisuutta. Toisaalta ei ole perusteltua olettaa tai väittää, että sosiaalinen paine muodossa tai toisessa aiheuttaisi ainoastaan negatiivisia asioita. Esimerkkinä tästä voidaan pitää jonkin yksittäisen työntekijän omiin tehtäviinsä kehittämää työkalua, joka halutaan ottaa laajempaan käyttöön. Tällöin sosiaalinen paine on hyväksi, jotta työkalu soveltuisi paremmin laajemmalle käyttäjäkunnalle.

Seuraavaksi esiteltävät kaksi OptoFidelityn työkalua ovat yrityksen ulkopuolelle myytäviä. Tästä johtuen käyttöliittymää on jalostettu eri versioiden myötä esimerkiksi käyttäjiltä saatujen palautteiden ja ehdotusten perusteella. Nykyisistä versioista käyttäjäkokemuksia kerättiin suppeilla avoimilla haastatteluilla, joissa OptoFidelity haastatteli nykyisten versioiden käyttäjiä. Nämä työkalut esitellään tutkimuksessa sen vuoksi, että ne on kohdennettu ammattilaisten käyttöön yritysten sisällä. Tämän vuoksi ne edustavat insinöörityökaluja eivätkä loppukäyttäjille suunnattuja tuotteita.

6.1 Watchdog

WatchDog-työkalulla käyttäjä voi mitata suurnopeuskameran, mikrofonin ja tarkkuusvaa'an avulla tarkastelun kohteena olevan laitteen käyttöliittymän ja loppukäyttäjän välisiä vuorovaikutustapahtumien kestoaikoja. WatchDogin käyttöä voidaan avata esimerkillä, jossa tarkoituksena on mitata käyttäjän ikonin painal-

luksen aiheuttaman käyttöliittymäreaktion kestoaikaa. Tällaisessa esimerkissä mitatun ajan keston aloituspiste on käyttäjän painallus tai painalluksen loppuminen riippuen siitä, kumpi aloittaa käyttöliittymällä näkyvän prosessin. Tämän aloituspisteen rekisteröinti WatchDogissa tapahtuu tarkkuusvaa'an avulla, joka kykenee tunnistamaan painalluksen sekä sen loppumisen. Mittauksen loppupisteen määrittäminen käyttöliittymän grafiikan osalta voidaan tehdä kahdella eri tavalla. Ensimmäinen tapa olisi hyödyntää työkalun Mask-toiminnallisuutta, jossa työkalu opetetaan tunnistamaan haluttu tila ja sen avulla työkalu kykenee laskemaan automaattisesti keston pituuden. Toisena tapana on käyttäjän suorittama kuva kuvalta tapahtuva tarkastelu ja manuaalinen kestoajan lasku. Tämä jälkimmäinen tapa on tarpeen, mikäli käyttöliittymällä tapahtuva reaktio ei ole selkeä tai vakio, jolloin koneen opettaminen tunnistamaan tapahtuma on työlästä.



Kuva 11: WatchDogin konfigurointinäkymä.

Ylläkuvatun esimerkin yksinkertaisimman mittaustapahtuman, jossa käyttäjä suorittaa mittauksen ilman Mask-toiminnon hyödyntämistä, askeleet ovat seuraavat:

1. Käynnistetään WatchDog-ohjelma konfiguraationäkymään (kuva 11).

2. Käyttäjä asettaa mitattavan kohteen tarkkuusvaa'alle siten, että kohde on sopivasti kameran näkökentässä.

3. Käyttäjä tarkentaa ja zoomaa kameran siten, että käyttöliittymän tarkasteltava kohta on selvästi WatchDog-ohjelman videoruudussa nähtävissä.

4. Käyttäjä tekee tarvittavat muutokset asetuksiin. Näitä mahdollisia asetuksia ovat seuraavat:

   (a) Kameran näyttämän kuvan katselukulman säätö tai valinta.

   (b) Sisäisen tapahtuman minimikeston määritys (Minimum intraevent duration).

   (c) Valotuksen säätö.

   (d) Tapahtuman havaitsemisen kynnyksen määritys (Event detection threshold).

   (e) Mittauksen aloituspisteen määritys (IO activation).

   (f) Automaattisen mittausraportin avaamisen päälle/pois-valinta.

5. Mittaustapahtuma aloitetaan painamalla ruudulla näkyvää "Go To Measurement Mode"-painiketta. Tämän jälkeen ohjelma jää odottamaan käyttäjän "Test start"-painikkeen painallusta, tarkkuusvaa'an tai mikrofonin antamaa signaalia.

6. Mittauksen alettua ohjelman tilaa ilmaiseva palkki oikeassa alanurkassa muuttuu vihreäksi ja teksti vaihtuu "Idle"-tekstistä "Analyzing"-tekstiin.

7. Käyttäjä voi lopettaa mittaustapahtuman painamalla ohjelman ruudulla näkyvää "Test stop"-painiketta.

Tämän lisäksi käyttäjä voi syöttää muistiinpanon mittaustapahtumasta ohjelman ruudulla näkyvään laatikkoon.

6.1.1 Tunnistetut ongelmat

WatchDogin käyttäjien haastatteluista ei noussut esille kriittisiä käytettävyysongelmia, jotka olisivat aiheuttaneet käyttäjille ylitsepääsemättömiä haittoja. Haastattelujen perusteella myös työkalun perustoiminallisuuden oppimiskäyrä ei ollut mahdottoman oloinen ilman käyttöohjettakaan. Haastatteluissa tuli kuitenkin esiin käytettävyyteen liittyviä parannusehdotuksia, joista osa liittyi ohjelmiston perustoiminallisuuteen ja osa työkalun käyttöliittymään. Haastatteluista ilmitulleet käyttöliittymään liittyvät parannusehdotukset:

1. Kameran valkotasapainon ja tarkkuuden säätäminen. Nämä asetukset tehdään kameran optiikkasäätimiä käyttämällä. Kuvan muutosta puolestaan seurataan WatchDogin ruudulta. Käyttäjän tavoitteena on saada WatchDogin näytöllä näkyvä mitattavan laitteen tai kohteen haluttu kohde selvästi näkyviin. Ohjelmisto ei auta kuvan säätämisessä käyttäjää.

2. Laitekohtaisen konfiguraation tallennus. Tässä työssä esiteltävässä WatchDogin versiossa (vm. 2013) tätä ominaisuutta ei ole täydellisesti toteutettu. Tämä osittain johtuu siitä, että kameran zoom-, tarkennus- ja valkotasapainosäädöt on tehtävä käsin.

3. Käyttöliittymän painikkeiden selkeys. Ongelman juuret ovat WatchDogin modaalisuudessa, jonka vuoksi eri painikkeiden roolit eri tiloissa muuttuvat. Varsinaista haittaa epäselvyyden lisäksi aiheutti tässä versiossa "Test stop"-painikkeen muuttuminen "Flush buffer"-painikkeeksi, jolloin mittausta pysäyttäessään käyttäjä voi helposti painaa vahingossa "Flush buffer"-painiketta. "Flush buffer"-painikkeen painaminen tässä tapauksessa aiheuttaa virhedialogin ilmestymisen keskelle WatchDogin ruutua.

4. Päällekkäiset "Exit"-painikkeet. Windows-käyttöjärjestelmän ikkunointi tarjoaa ohjelmalle käyttöjärjestelmän oman yhtenäisen "sulje/lopeta"-painikkeen ikkunan yläkulmasta. WatchDogin käyttöliittymän estetiikka ja selkeys kärsii hieman, kun saman toiminnon omaava WatchDogin painike tuodaan lähelle käyttöjärjestelmän omaa nappia.

5. Virheilmoitusten epäselvyys. WatchDogin virheilmoitukset ovat koodipohjaisia. Esimerkkinä voidaan mainita ohjelmiston "Error 4"-virheilmoitus, joka ei ilman lisäinformaatiota kerro käyttäjälle mitään muuta kuin, että virhe on tapahtunut.

6. WatchDogin tuottama mittausraportti ei sisällä kaikkea, mitä laitteen ruudulla on tapahtunut. Käyttäjän saama mittausraportti esittää mittausdataa, jossa on esitetty havaittujen muutoksien välistä aikaa. Haastattelussa käyttäjä ei mielestään nähnyt raportista kaikkea, mitä paljaalla silmällä laitteen ruudulta oli nähtävissä.

7. Valikot ovat piilossa. WatchDogissa joidenkin ruudulla olevien painikkeiden taakse on sijoitettu toiminallisuutta. Tätä toiminallisuutta ei tuoda käyt-

täjälle esiin ja sen havaitseminen silmämääräisesti on mahdotonta.

8. "Flush Buffer"-valinta vahingossa. WatchDogissa kyseisen toiminnon esiintuominen on toteutettu siten, että se tulee näkyville käyttäjän painettua "Test stop"-painiketta. Tuntemattomasta syystä ohjelmisto saa käyttäjän antaman yhden hiiren napin painalluksen kahtena, josta seuraa "flush buffer"-toiminto ketjun ensimmäinen askel, jossa kysytään varmennusta toiminnon suoritukselle. Tahaton painaminen voi myös syntyä tilanteesta, jossa WatchDog-ohjelmistoa ajava kone on ylikuormitettu, ja ohjelmisto ei kykene antamaan välitöntä palautetta käyttäjälle ensimmäisestä painalluksesta, mikä puolestaan johtaa käyttäjän toiseen painallukseen.

9. "Mask In Use"-teksti harhaanjohtaa käyttöliittymässä. WatchDogin käyttöliittymän yläosassa on teksti "Mask In Use" ja sen perässä indikaattori, joka ilmaisee sen, onko kyseinen ominaisuus aktiivisena. Haastatteluista selvisi, että tämä teksti johtaa käyttäjää harhaan. Tämä siksi, koska tekstin muotoilu on sellainen, että siitä muodostuu käyttäjälle kuva, että ominaisuus on päällä.

10. Ohjelmiston tuottamien tuloksien toistettavuus. Haastatteluista paljastui, että mittaustulosten toistettavuus ei välttämättä ole aina niin varmaa kuin sen pitäisi olla. Tämä johtuu haastateltavien mielestä ohjelmistosta löytyvistä lukuisista säädöistä, joilla on vaikutusta mittaustuloksiin.

6.1.2 Tunnistettujen ongelmien lyhyet korjausehdotukset

1. Käytettävyyden näkökulmasta olisi hyvä toteuttaa koko säätöprosessi mahdollisimman automaattiseksi. Tämä vaatisi suurnopeuskameraan motorisoidut säätimet, joiden avulla käyttäjän käsin tekemä säätö muuttuisi tarpeettomaksi. Lisäksi voidaan olettaa, että kone kykenisi tekemään säädön käyttäjää nopeammin. Tämän edellytyksenä tosin on se, että kameraa säätävän ohjelmiston tulisi olla tietoinen halutusta kohteesta. Vähäisillä kustannuksilla tilannetta voisi parantaa ohjelmistossa tekemällä indikaattorit käyttöliittymälle, joiden avulla käyttäjälle välittyisi tieto siitä, kuinka tarkasti ohjelmisto kykenee näkemään kameran näkökentässä olevat asiat.

2. Ongelma on riippuvainen ensimmäisen kohdan ratkaisusta, koska automaattista tai avustettua tarkennusta tarvitaan tallennettujen laitekonfiguraatioiden käyttöönoton yhteydessä.

3. Painikkeiden roolien vaihtelun ollessa ongelma yhtenä ratkaisuna voisi olla toimintojen tuominen omiksi painikkeiksi. Haastattelujen perusteella käyttäjien mielestä käyttöliittymässä on painikkeita, joita ei juurikaan käytetä. Nämä turhat painikkeet voisi poistaa ja siten luoda tilaa enemmän käytössä olevien toimintojen omille painikkeille. Toisena ratkaisuna on selkeyttää painikkeiden roolin vaihtumista selkeämmillä symboleilla tai väreillä. Esimerkkinä voidaan mainita vaikka "Test start"-painikkeen muuttuminen "Test stopiksi". Värikoodauksella painikkeen roolin vaihtuminen voisi olla selkeämpää.

4. Ratkaisuehdotus tähän ongelmaan on yksinkertainen ylimääräisen painikkeen poisto.

5. Virheilmoitusten epäselvyyteen korjausehdotus on luonnollisesti niiden selkiyttäminen ja käyttäjälle relevantin tiedon antaminen. Tätä varten ohjelmistoon pitäisi toteuttaa oman tilan tarkkailu, joka selkeyttäisi vian perimmäistä syytä. Tämän tietouden avulla myös ohjelmiston omaa vikasietoutta ja automaattista vikatilanteesta toipumista voitaisiin kehittää.

6. Mittausraportin sisältöön ei ole mitään kaiken kattavaa korjausehdotusta, koska sen sisällön eri elementtien tarpeellisuus vaihtelee tapauskohtaisesti. Periaatteessa mittausraporttiin on saatavilla kaikki mittaustapahtuman kuvat, koska ne on tallennettu mittaustapahtuman hakemistoon. Yhtenä helpotuksena ongelmaan raporttiin voisi liittää aikajanan, jonka avulla käyttäjä voisi tutkia tallennettuja kuvia, vaikka itse raportointityökalu ei olisikaan löytänyt kuvista mitään raportoinnin arvoista.

7. Ongelman yhtenä ratkaisuna on yhtenäistää aktiivisten kontrollikomponenttien ulkoasua siten, että ne erottuvat kiinteistä käyttöliittymän komponenteista. Tämän voisi toteuttaa esimerkiksi muuttamalla aktiivisten painikkeiden värit yhtenäiseksi ja valitsemalla käytetty väri siten, että se poikkeaa muista käyttöliittymän komponenteista.

8. Ratkaisuehdotuksena “Flush buffer”-painikkeen kohdalla selkeyden vuoksi olisi erottaa se omaksi painikkeekseen. Tutkimalla kyseisen toiminnon käyttöä voitaisiin saada myös selville, olisiko siitä järkevää tehdä käytettävyyden tehokkuuden näkökulmasta yhdistelmäpainike. Oletettavaa on, että “Flush buffer”-toimintoa käytetään, kun mittaustapahtumassa menee jotain vikaan. Tällöin käyttäjän voidaan olettaa ensin painavan “test stop”-painiketta ja sen jälkeen “Flush buffer”-painiketta analyysin odottamisen välttämiseksi ennen seuraavaa yritystä. Tälläisessä tilanteessa “test stop”- ja “Flush buffer”-toimintojen yhdistäminen erillisen painikkeen taakse voisi tehostaa käyttäjän toimintaa.

9. “Mask In Use”-tekstin hämätessä haastateltavia loogisena ratkaisuehdotuksena on muuttaa tekstiä siten, että se muuttuu tilanteen mukaan. Erillisen värillisen indikaattorin voisi tällöin poistaa.

10. Tämän ongelman ratkaisu riippuu osittain ensimmäisen ja toisen ongelman ratkaisuehdotuksista. Käyttäjän tehokkuutta tässä ongelmassa voitaisiin ratkaista tarjoamalla työkalu, jonka avulla käyttäjä voisi verrata tekemäänsä mittausta johonkin aikaisempaan mittaukseen. Tämän tiedon avulla käyttäjän olisi mahdollista löytää syy mittaustulosten eroavaisuuteen.

6.1.3 GOMS-mallin soveltaminen WatchDogin käyttöliittymään

WatchDog-ohjelmiston kohdalla GOMS-malli tarjoaisi tulevien käyttöliittymäsuunnitelmien arviontiin soveltuvia menetelmiä jo ennen varsinaista suunnitelman toteuttamista ohjelmaan. Ohjelmistolla suoritettavat tehtävät soveltuvat rakenteeltaan GOMS-mallin menetelmille ja siksi, mallin kanssa voisi saavuttaa käyttökelpoista tietoa. Koska WatchDog-ohjelmistoa on myyty eri yrityksille, on oletettavaa, että sen käyttötapoja on useita erilaisia. On hyvin todennäköistä, että yksi käyttöliittymäversio ei palvele jokaista eri käyttötapaa yhtä tehokkaasti. GOMS-malli tarjoaisi yhden tavan parantaa käyttöliittymän ja käyttäjän välistä vuorovaikutusta ja siten parantaa tehtävien suorituksen tehokkuutta.

Mallin käyttö olisi kuitenkin järkevää vasta sitten, kun suunnittelijoilla on tarkka käsitys käyttäjien tarpeista. Tämän käsityksen voisi saavuttaa seuraamalla vanhojen käyttäjien työskentelyä ja työskentelytapoja ohjelmiston nykyisellä versiolla. Uusien potentiaalisten käyttäjien kohdalla vastaavaa tietoutta voisi kerätä

kartoittamalla heidän tarpeensa ja vertaamalla niitä tarpeita olemassa olevaan tietouteen vastaavista tapauksista.

6.1.4 Esimerkkitapauksen analysointi

WatchDog-ohjelmiston yksi peruskäyttötapaus on mitata kosketusnäytön koskетuksen ja laitteen antaman palautteen välistä aikaa eli niin sanottua vasteaikaa. Tällainen mittaus käsin tehtynä ilman robottiavustusta vaatii mittausta suorittavalta henkilöltä vuorovaikutusta itse mittauskohteena olevan laitteen kanssa sekä WatchDog-ohjelmistoa ajavan tietokoneen kanssa. Tällainen mittaustapahtuma voidaan jakaa GOMS-mallin mukaisesti pienempiin kokonaisuuksiin eli välitavoitteisiin. Välitavoitteet tällaisessa esimerkkitapauksessa ovat seuraavat, mikäli oletetaan WatchDog-ohjelmiston olevan jo valmiiksi käynnissä ennen tehtävän aloittamista:

1. Mitattavan laitteen valmistelu mittausta varten siten, että laite on käyttäjän syötettä odottavassa tilassa.

2. WatchDog-ohjelmiston saattaminen valmiustilaan mittaustapahtumaa varten. Tämä tapahtuu painamalla käyttöliittymässä olevaa “Test Start”-painiketta.

3. Syötteen antaminen laitteelle ja halutun reaktion odottaminen.

4. Mittauksen lopettaminen “Test stop”-painiketta painamalla WatchDogin käyttöliittymässä.

5. Mittaustuloksen analysointi.

Kukin kohta ylläolevasta listauksesta voidaan purkaa toimilistaksi, josta nähdään kuhunkin välitavoitteeseen kuuluvat toimet. Toimet ylläoleviin välitavoitteisiin voisivat olla seuraavat:

- Mitattavan laitteen valmistelu. Välitavoitteen kokonaisaika: 13 sekuntia.

    1. Laitteen asettaminen kameran alle mittausalustalle (5 s).
    2. Kameran kohdistus haluttuun kohteeseen (5 s).
    3. Laitteen käyttöliittymän asettelu haluttuun tilaan mittausta varten (3 s).

- WatchDog-ohjelmiston valmistelu mittausta varten. Välitavoitteen kokonaisaika: 4 sekuntia.

    1. Asetuksien pikasilmäys (3 s).
    2. Ohjelmiston siirtäminen mittaustilaan painamalla “Go To Measurement Mode”-painiketta (1 s).

- Mittaustapahtuma. Välitavoitteen käyttäjältä vaadittavien toimien kokonaisaika: 1,5 sekuntia.

    1. Laitteen näytön kosketus (0,5 s).
    2. Halutun transaktion odottaminen.
    3. “Test stop”-painikkeen painaminen mittauksen lopettamiseksi (1 s).

- Tuloksien tarkistus. Käyttäjältä vaadittavien toimien kokonaisaika: 5 sekuntia.

    1. Mittausdatan odottelu (riippuu mittaustapahtuman pituudesta).
    2. Aukeavan raportin selaus haluttuun kohtaan (riippuu havaittujen muutosten määrästä).
    3. Tuloksen kirjaus (5 s).

Mittaamalla käyttäjältä vaadittaviin toimiin kuluvaa aikaa saadaan hyvinkin yksityiskohtaista tietoa tehtävävään kulutetusta ajasta. Tätä tietoa hyödyntämällä voidaan arvioida samaan välitavoitteisiin pyrkivien metodien välisiä eroja.
GOMS-mallin käyttöä WatchDogin käyttöliittymän tehostamisessa voidaan havainnollistaa jatkamalla esimerkkitapausta käsittämään laajempaa mittaustapahtumaa. Esimerkkinä tälläisestä mittaustapahtumasta WatchDogilla on kosketusnäytöllisen laitteen loppukäyttäjän kokemien viiveiden mittaaminen. Tällainen mittaustapahtuma koostuu yleensä ennaltamääriteltyjen suorituskykyilmaisimien (KPI=Key Performance Indicator) mittaamisesta. Suorituskykyilmaisimen mittaamisessa hyväksi todettu tapa on mitata sama tapaus monta kertaa ja ottaa kertyneistä tuloksista mediaani. Yksittäinen mittaustapahtuma tälläisessa tapauksessa voi olla aikaisemmin mainitun käyttöesimerkin kaltainen. Erona yksittäiseen mittaustapahtumaan on vain se, että tapahtumasarjaa toistetaan useampi kerta.

Ylläolevasta listauksesta voi jo pikaisella silmäyksellä havaita mahdolliset kehityspaikat, kun samaa tehtävää on tarkoitus tehdä monta kertaa peräkkäin. Yhtenä tälläisenä voidaan tutkia kahta viimeistä välitavoitetta eli mittaustapahtumaa ja tuloksien tarkistamista. Nykyisellä toteutuksella saman testin toistaminen peräjälkeen voi tapahtua kahden hieman erilaisen tapahtumaketjun kautta. Molempien tapahtumaketjujen askeleet WatchDogin käyttöoppaan testin suoritusesimerkkiä [OptoFidelity Oy, 2014, s. 16] mukaillen ovat seuraavat alkuvalmistelujen jälkeen, kun ohjelmisto on valmiina "Go To Measurement mode"-painikkeen painalluksen jälkeen:

1. Aloita testi painamalla "Test Start"-painiketta tai käytä automaattista käynnistystä (vaaka tai mikrofonin syöte).

2. Tarkkaile testin edistymistä.

3. Pysäytä testi painamalla "Test Stop"-painiketta.

4. Odota testidatan analyysin valmistumista.

Tämän jälkeen käyttäjä voi päättää, haluaako hän tarkastaa tämän vasta tehdyn testin tulokset vai suorittaako lisää testikierroksia ennen tuloksien tarkastelua. Tämä valinta haarauttaa tapahtumaketjut toisistaan. Tulosten tarkastelun tässä tapauksessa voidaan olettaa tapahtuvan WatchDogin tarjoaman "Event Report"-näkymän avulla. Tämä raportti avataan automaattisesti testin analyysin valmistutttua tai manuaalisesti testidatahakemistosta. Raportti avautuu erilliseen selaimen ikkunaan. Käyttäjä, joka haluaa tarkastaa tuloksen yksitellen jokaisen testikierroksen jälkeen, tekee ylläolevien askelten lisäksi seuraavat lisäaskeleet testikierrosten välissä:

- Aukaisee testiraportin, tai odottaa automaattista aukaisua testiraportille.

- Lukee tuloksen raportista.

- Kirjaa tuloksen ylös.

- Palaa takaisin WatchDog-ohjelman ikkunaan selaimen ikkunasta.

- Painaa "Go To Measurement Mode"-painiketta.

Käyttäjä, joka tarkastaa tulokset vasta, kun kaikki testikierrokset on tehty, testin analyysin valmistuttua painaa WatchDogin ikkunassa “Go To Measurement Mode”-painiketta aloittaakseen uuden testikierroksen. Kierrosten loputtua käyttäjä tekee seuraavat askeleet tulosten tarkastamiseksi:

- Avaa selaimen tai siirtyy selaimen ikkunaan, mikäli selain on jo valmiiksi käynnissä.
- Avaa selaimella testiraportin.
- Lukee tuloksen raportista.
- Kirjaa tuloksen ylös.

Tämä sama askelsarja toistetaan jokaiselle tehdylle testille. Edellämainitut tapahtumaketjut ovat verrattain suoraviivaisia ja eivät juurikaan mahdollista käyttäjän suorittamaa rinnakkaisajoa. Tästä syystä näiden kahden erilaisen tapahtumaketjun vertaaminen KLM-mallin avulla on mahdollista. Käyttäjän tekemä rinnakkaisajo olisi käytännöllisesti mahdollista ainoastaan pitempikestoisien testien yhteydessä, jolloin käyttäjä voisi tarkastella ja kirjata edellisen kierroksen tulosta uuden testikierroksen ajon aikana. Tälläisessä tapauksessa muiden GOMS-mallien käyttö KLM-mallin sijaan on suositeltavaa. KLM-mallin käyttöä varten määritellään tähän tilanteeseen sopivat operaattorit:

Operaattori	Selite	Kesto(s)
K	Kosketusnäytön painallus	0.1
B	Hiiren napin painaminen/vapautus	0.1/0.2
P	Kohteen osoittaminen näytöllä	1.0
R	Tulokset lukeminen raportista	5.0
T	Tuloksen kirjaaminen	5.0
H	Käsien siirto näppäimistön tai hiiren päälle	0.4
W	Järjestelmän palautteen odottaminen	5.0
M	Henkinen valmistautuminen	1.2

Taulukko 9: Esimerkkitapauksen operaattorit karkeine kestoaika-arvioineen.

KLM-mallin käyttöä varten tehdään seuraavat oletukset:

- Käyttäjä pystyy koskettamaan testattavan laitteen näyttöä toisella kädellä samaan aikaan, kun toinen käsi on edelleen hiiren päällä.
- Testattava asia ei ole pitkäkestoinen, jolloin käyttäjällä ei ole rinnakkaisajon mahdollisuutta.
- Käyttäjällä on automaattinen testin aloitus vaa'an tai mikrofonin syötteen avulla käytössä.
- Käyttäjällä on käytössä raportin automaattinen avaus, mikäli tuloksia tarkastellaan jokaisen kierroksen jälkeen.
- Analyysin kohteena oleva kierros ei ole viimeinen.
- Testikierroksia on useita kappaleita.

Ensimmäinen mahdollinen tapahtumaketju, jossa tulokset tarkastetaan ja kirjataan jokaisen testikierroksen jälkeen, KLM-mallin mukaisesti esitettynä on seuraavanlainen:

Toimi	Operaattori	Kesto(s)
Testin aloitukseen valmistautuminen	M	1.2
Käden siirto testilaitteelle	H	0.4
Hiiren kursorin vienti "Test Stop"-painikkeen päälle	P	1.0
Reaktion aiheuttaminen kosketusnäytöllä	K	0.1
Testin loppumisen odotus	W	3.0
"Test Stop"-painikkeen klikkaus	BB	0.2
Analyysin loppumisen odotus	W	5.0
Valmistautuminen tuloksen lukuun	M	1.2
Tuloksen luku raportista	R	5.0
Tuloksen kirjaaminen	T	5.0
Valmistautuminen aktiivisen ikkunan vaihtoon	M	1.2
Hiiren kursorin vienti WD-ikkunan päälle	P	1.0
WD-ikkunan aktivointi klikkaamalla	BB	0.2
Hiiren kursorin vienti "Go To Measurement Mode"-painikkeelle	P	1.0
Painikkeen klikkaus	BB	0.2
	Yhteensä:	25.7 s.

Taulukko 10: KLM-mallin mukainen tahtumalistaus.

Toisessa tapahtumaketjussa, jossa tulokset luetaan testikierrosten jälkeen, on tarpeen tehdä kaksi listausta. Ensimmäisessä listataan mittaustapahtuma ja toisessa tuloksen luku.

Toimi	Operaattori	Kesto(s)
Testin aloitukseen valmistautuminen	M	1.2
Käden siirto testilaitteelle	H	0.4
Hiiren kursorin vienti “Test Stop”-painikkeen päälle	P	1.0
Reaktion aiheuttaminen kosketusnäytöllä	K	0.1
Testin loppumisen odotus	W	3.0
“Test Stop”-painikkeen klikkaus	BB	0.2
Analyysin loppumisen odotus	W	5.0
Valmistautuminen uuteen kierrokseen	M	1.2
Kursorin vienti “Go To Measurement Mode”-painikkeelle	P	1.0
Painikkeen klikkaus	BB	0.2
	Yhteensä:	13.3 s

Taulukko 11: Testien suorituksen KLM-mallin mukainen esitystapa.

Testikierrosten jälkeen käyttäjän on aktivoitava selainikkuna, joka vaatii seuraavat toimet:

Toimi	Operaattori	Kesto(s)
Valmistautuminen selaimen ikkunan aktivointiin	M	1.2
Kursorin vienti selainikkunan päälle	P	1.0
Ikkunan aktivointi klikkaamalla	BB	0.2
	Yhteensä:	2.4 s

Taulukko 12: Toimet.

Tämän jälkeen käyttäjän on nykyisen WatchDogin toteutuksen mukaan avattava jokainen raportti yksitellen. Tämä voi tapahtua selaimen omien toimintojen kautta tai erillisen tiedostonhallintaohjelman kautta. Tässä analyysissä oletetaan tulosten läpikäynnissä seuraavia asioita:

- Käyttäjä käyttää selaimen omaa työkalua tiedostojen avaamiseen.
- Tiedostoa aukaistaessa oletushakemistona on testidatahakemiston juuri, jonka alla on jokaiselle testille oma alihakemistonsa.
- Käytetty selain listaa hakemistosta vain *.html-päätteiset tiedostot.

- Käyttäjä osaa käyttää “ctrl-o”-näppäinyhdistelmää tiedoston valintadialogin käynnistämiseksi. Tämä siksi, koska monissa käyttöliittymäkonfiguraatioissa selaimilla ei ole erillistä työkaluriviä ylhäältä, josta tiedoston avaaminen löytyisi päätasoilta.

KLM-mallin mukaiset askeleet näillä oletuksilla seuraavat tulosten läpi käymiseksi yhden kierroksen osalta ovat:

Toimi	Operaattori	Kesto(s)
Valmistautuminen tuloksen avaamiseen	M	1.2
Käsien asettaminen näppäimistölle	H	0.4
“Ctrl-o”-näppäinyhdistelmän painallus	K	0.1
Käden siirto takaisin hiirelle	H	0.4
Hiiren kursorin siirto halutun hakemiston päälle	P	1.0
Hakemiston klikkaus	BB	0.2
Hiiren kursorin siirto raportin sisältävän tiedoston päälle	P	1.0
Halutun tiedoston klikkaus	BB	0.2
Valmistautuminen lukemaan tulos	M	1.2
Tuloksen luku	R	5.0
Tuloksen kirjaus	T	5.0
	Yhteensä:	15.7 s

Taulukko 13: Yhden kierroksen KLM-mallin mukainen esitystapa.

Tuloksia vertaamalla voi huomata, että jälkimmäinen tapahtumaketju ei ole nopeampi tapa, vaikka se saattaisikin tuntua nopeammalta tavalta pikaisesti tarkasteltuna. KLM-mallin arvion mukaan testin suoritukseen ja sen perään tuloksen käsittelyyn menee aikaa 25.7 sekuntia. Erillisinä kokonaisuuksina testin ja tuloksen käsittely KLM-mallin ennusteen mukaan vaatii ensimmäisen kierroksen kohdalla $13.3 + 2.4 + 15.7 = 31.4$ sekuntia. Ensimmäisen tuloksen tarkastuskierroksen jälkeen selainikkunan aktivontia ei tarvitse tehdä uudestaan ja silloin ennusteen mukainen aika-arvio yhdelle kierrokselle on 27.0 sekuntia. KLM-mallin analyysin tulosten valossa on perusteltua väittää, että nykyinen toteutus tukee testin ja tuloksen analysointia yhtenäisenä tapahtumaketjuna. Selkeänä indikaattorina tästä on käyttöliittymästä löytyvä “Auto open report”-toiminto, joka poistaa selaimen ja raportin avaamiseen kuluvan ajan. Ilman “Auto open report”-toimintoa tilanne kääntyisi toisen tapahtumaketjuvaihtoehdon hyväksi toisen testikierroksen jälkeen. Ero tällöin muodostuisi selainikkunan ja WatchDog-ikkunan välillä vaihteluun. Tapahtumaketjuja tarkastelemalla voi havaita kaksi

konkreettista kohtaa, joita parantamalla voitaisiin tehostaa peräkkäisten testien suorittamista. Näistä esimerkkeinä voisivat toimia seuraavat parannukset:

- Lisäämällä käyttöliittymään komponentti, joka antaa käyttäjän määritellä peräkkäisten testien määrän. Tämä toiminto tukisi erityisesti esimerkkinä käytetyn tilanteen mukaista mittaustapahtumaa, jossa samaa asiaa tulee mitata useampi kerta peräkkäin.

- Peräkkäisten mittausten raporttien yhdistäminen tai koostesivun koostaminen siten, ettei yksittäistä raporttia tarvitse avata selaimen tai käyttöjärjestelmän tiedostonhallinnan kautta.

- Testin loputtua tehtävän analyysin muuttaminen taustalla ajettavaksi prosessiksi. Tällä muutoksella vähennettäisiin käyttäjän kokemaa järjestelmän odotteluaikaa testikierrosten välillä. Lyhyempien testien kohdalla tämä voisi nykypäivän moniytimillisillä prosessoreilla tarkoittaa sitä, että käyttäjä joutuu odottamaan analyysin valmistumista ainoastaan viimeisen kierroksen jälkeen.

Ehdotettujen parannusten tuomaa hyötyä voidaan myös arvioida KLM-mallia käyttämällä. Aikaisempien oletusten ollessa voimassa yhden testikierroksen toimet ovat seuraavat viimeistä kierrosta lukuunottamatta:

Toimi	Operaattori	Kesto(s)
Testin aloitukseen valmistautuminen	M	1.2
Käden siirto testilaitteelle	H	0.4
Hiiren kursorin vienti “Test Stop”-painikkeen päälle	P	1.0
Reaktion aiheuttaminen kosketusnäytöllä	K	0.1
Testin loppumisen odotus	W	3.0
“Test Stop”-painikkeen klikkaus	BB	0.2
Valmistautuminen uuteen kierrokseen	M	1.2
Kursorin vienti “Go To Measurement Mode”-painikkeelle	P	1.0
Painikkeen klikkaus	BB	0.2
	Yhteensä:	8.3 s

Taulukko 14: Arvioidut suoritusajat parannusehdotuksien kanssa.

Viimeiselle kierrokselle summaan tulee lisätä oletettu 3.0 sekuntia analyysin valmistumisen odottamiselle. Testin tuloksien tarkastelun kohdalle parannusten jälkeen

voidaan olettaa seuraavat toimet, jos oletetaan, että raportit on nähtävillä yhdeltä koostesivulta ja tämä koostesivu avautuu “Auto open report”-toiminnon avulla viimeisen testikierroksen jälkeen automaattisesti:

Toimi	Operaattori	Kesto(s)
Valmistautuminen tuloksen avaamiseen	M	1.2
Hiiren kursorin siirto halutun testiraportin päälle	P	1.0
Raportin nimen tai ikonin klikkaus	BB	0.2
Tuloksen luku	R	5.0
Tuloksen kirjaus	T	5.0
	Yhteensä:	12.4 s

Taulukko 15: Tulosten tarkasteluun kuuluvat toimet.

Viiden testikierroksen kanssa aika-arvio olisi seuraava parannusten kanssa:

Toimikokonaisuus	toistokerrat	Kulunut aika(s)
Testiajot	5	41.5
Viimeisen analyysin odotus	1	5
Tulosten tarkastus ja kirjaus	5	62
	Yhteensä:	108 s

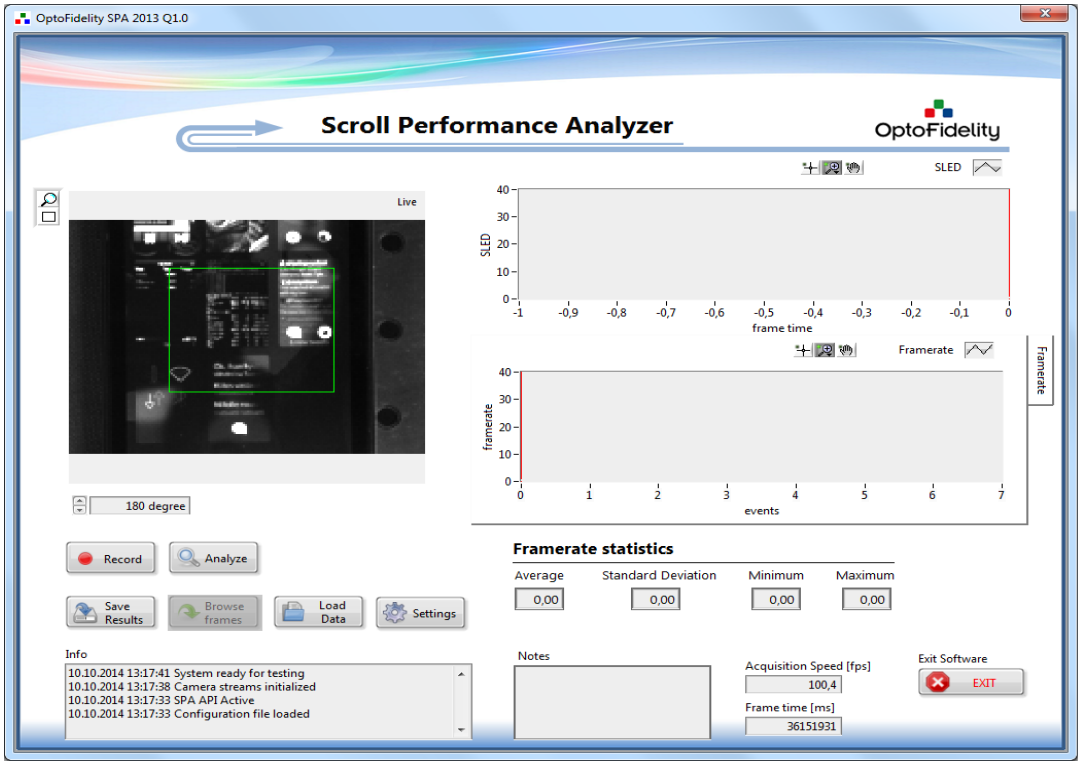
Taulukko 16: Kokonaisaika-arvio.

Nykyisellä toteutuksella viiden testikierroksen teko siten, että testin suoritus ja tuloksen tarkastus ovat yhtä kokonaisuutta, vaatii analyysin perusteella aikaa: 5 x 25.7 s = 128.5 sekuntia. Ero edellä olevien oletuksien mukaan olisi siis parannusten ja uuden suoritusmetodin kanssa noin 20 sekuntia parempi. Eron määrään vaikuttaa oleellisesti testin pituus ja käytettävän kameran resoluutio sekä kuvanottonopeus. Nämä seikat vaikuttavat oleellisesti testiajon jälkeen tehtävän analyysin pituuteen. Saavutettu hyöty kertautuu testattavien asioiden lukumäärän mukaan. Loppukäyttäjälle suunnitellun normaalin tuotteen testilistassa tällaisia kohtia saattaa olla 50. Arvioidun 20 sekunnin hyödyn kanssa 50-kohtaisen testilistan kanssa kyse on jo huomattavasta aikaerosta, kun lasketaan 20 s x 50 = 1000 sekuntia.

6.2 SPA

SPA:n (Scroll Performance Analyser) päätarkoitus on mitata mittauskohteena olevan laitteen näytön ruudun päivitysnopeutta (FPS) halutuissa laitteen toiminnoissa. Mittaaminen tapahtuu suurnopeuskameraa hyväksikäyttäen. SPA:an toimintaa voidaan avata esimerkillä, jossa käyttäjän tarkoituksena on mitata laitteen

internet-selaimen ruudunpäivitysnopeutta, kun sivua vieritetään johonkin suuntaan. Mittaustapahtumaa varten käyttäjän tulee asettaa mittauskohteena toimiva laite suurnopeuskameran alle siten, että SPA:an näytöllä olevasta ruu-dusta näkyy mittauskohteena oleva ruutu halutulta osin. Tämän jälkeen käyttäjän tulee määrittää varsinainen mittauskohde SPA:n ruudulta venyttämällä suorakulmio halutun kohteen ympärille (kuva 12). Käyttäjän tulee huomioida, että tällä määritetyllä alueella ei tulisi näkyä mitään muuta. Käytännössä tämä tarkoittaa kosketusnäytöllisessä laitteessa käyttäjän sormea tai robotin tekosormea, jolla selaimen näkymän vieritys tehdään. Itse mittauksen aloittaakseen käyttäjän tulee painaa käyttöliittymällä olevaa “Record”-painiketta, joka käynnistää SPA:an videon tallennuksen. Videon tallennuksen alettua käyttäjä vierittää selaimen ruutua. Vierittämisen jälkeen käyttäjän tulee painaa “Recording”-painiketta lopettaakseen videon tallennuksen. Videon tallennuksen päätyttyä ohjelmisto automaattisesti ryhtyy analysoimaan ruudunpäivitysnopeutta talletetusta videosta. Analysoinnin valmistuttua käyttäjä voi tarkentaa analysoitavaa kohtaa käyttöliittymän ylemmästä graafista. Tarkennus on tarpeen, koska muuten näkyvillä olevaan tulokseen saattaa sisältyä ajankohtia mittaustapahtuman alusta ja lopusta, jolloin käyttäjä ei vieritä ruutua ja sen vuoksi ruutu ei päivity.



Kuva 12: SPA:n aloitusnäkymä.

6.2.1 Tunnistetut ongelmat

Haastattelujen perusteella käyttäjät eivät tuoneet esiin mitään kriittisiä käytettävyysongelmia ohjelmistossa. Monet pienemmät ongelmat, jotka liittyvät käytettävyyteen tai toiminnallisuuteen olivat hyvin pitkälle samanlaiset kuin WatchDogillakin. Perusmittausten kohdalla oppimiskäyrä SPA:ssa oli WatchDogin tapaan hyvin matala ilman käyttöohjettakin.

1. Kameran valkotasapainon ja tarkkuuden säätäminen. Tämä ongelma on hyvin pitkälle samanlainen kuin WatchDog-ohjelmistossakin. SPA:n tapauksessa ruudunpäivitysnopeuden mittaus vaatii tarkemmat säädöt siihen, että konenäköelementti kykenee erottamaan ruudulla tapahtuvat muutokset riittävän selkeästi. Mittaustuloksien luotettavuus kärsii huonoista säädöistä ja siksi niiden säätäminen kohdalleen on hyvin tärkeää. Käyttäjällä säätöjen varmistukseen yksi keino on kokeilu ja silmämääräinen säätö. Tällä menetelmällä voidaan saavuttaa kelvolliset asetukset, mikäli peräkkäisissä mittauksissa tulokset ovat keskenään sopusoinnussa.

2. Käyttöliittymän painikkeiden selkeys. Haastatteluissa tämäkin ongelma on yhteinen WatchDogin kanssa. Esimerkkinä yhtenä epäselvänä painikkeena voidaan mainita käyttöesimerkin kohta, jossa käyttäjä lopettaa nauhoituksen painamalla “Recording”-painiketta. Käsitteellisesti selkeämpi nimi painikkeelle voisi olla “Stop recording”.

3. Kaksi päällekkäistä “Exit”-painiketta. Ongelma on hivenen erilainen SPA:n käyttöliittymällä kuin WatchDogilla, mutta periaatteeltaan samanlainen.

4. Virheilmoitusten epäselvyys. Ongelma on yhteinen WatchDogin kanssa. Selkokielisemmät virheilmoitukset tukisivat käyttäjää virhetilanteesta toipumisessa.

5. Tehdyn mittauksen häviäminen. Haastattelujen perusteella tämä ongelma esiintyy varsinkin, kun käyttäjä suurentaa näkymää johonkin hetkeen aikaskaalalla ja painaa “Analyze”-nappia. Tämän jälkeen tehty mittaus häviää ja tuloksen saamiseksi testaaja joutuu aloittamaan mittauksen alusta. Tämä voi olla kiusallinen ongelma, jos jonkin harvoin toistuvan ongelman mittaustieto häviää, eikä tilannetta saada toistettua uudestaan.

6. Valikot ovat piilossa. SPA:n ruudulla olevien painikkeiden taakse on piilotettu toiminnallisuutta samaan tapaan kuin WatchDogissakin. Silmämääräisesti näiden toimintojen löytäminen on hyvin vaikeaa, ja käytännössä ilman ohjeita käyttäjä voi löytää ne vain klikkailemalla kaikkea ruudulla näkyvää eri tilanteissa.

7. Ohjelmiston tuottamien tuloksien toistettavuus. Ongelma on samantapainen kuin WatchDogilla, mutta esiintyy useammin varsinkin tapauksissa, joissa ruudun vieritystä tehdään käsin robottisormen sijaan. Tällöin eri mittaajien tuottamissa tuloksissa voi olla suuriakin eroja. Myös mittaajan määrittämä kohta, mistä SPA mittaa ruudunpäivitysnopeuden, vaikuttaa. Huolimaton käyttäjä voi tehdä myös helposti virheen valitsemalla liian ison kohdealueen, jolloin käyttäjän oma sormi kulkee alueen yli. Tällöin tulos on yleensä virheellinen, koska näytön pikselien sijaan ohjelmisto on laskenut sormen liikkeen nopeutta näyttöä vasten.

8. Näytetyn kuvan koko SPA:n näytöllä. Haastateltavat kokivat, että SPA:n käyttöliittymän ruutu, josta näytetään kohteen videokuvaa, voisi olla suurempi WatchDogin käyttöliittymän tapaan. Lisää tilaa haastateltavat isommalle ruudulle loisivat poistamalla käyttöliittymästä vähemmän käytettyjä komponentteja.

6.2.2 Tunnistettujen ongelmien lyhyet korjausehdotukset

1. WatchDogin korjausehdotus voisi oletettavasti toimia myös SPA:n kohdalla. Tarjoamalla indikaattorit halutun kohteen liikkeen tunnistuksesta käyttäjä välttyisi varsinaisilta koekäyttökierroksilta, jolloin supistettaisiin säätöön kuluvaa aikaa. Täysin automaattisen ratkaisun kohdalla vaadittaisiin WatchDogin ehdotuksen lisäksi robottisormi tai kohdelaitteen käyttöjärjestelmään tehtävä muutos, joka aiheuttaisi ruudunpäivitystä ruudulla tuottamaan dataa tarkimpien mahdollisten asetusten hakuun.

2. Painikkeiden selkeyttä voisi parantaa WatchDogin tapaan värikoodaamalla aktiiviset komponentit siten, että ne erottautuvat selkeästi muista käyttöliittymän komponenteista. Selvän erottelun avulla käyttäjällä ei olisi vaikeuksia hahmottaa painikkeiden rooleja.

3. Turhan “Exit”-painikkeen poisto.

4. Korjausehdotus virheilmoitusten sisällön epäselvyyteen on samantapainen kuin WatchDogin kohdalla.

5. Mittausdatan automaattinen välitallennus ennen seuraavaa toimenpidettä voisi helpottaa mittausdatan häviämisongelmassa. Välitallennuksen avulla käyttäjällä olisi mahdollisuus palata edelliseen pisteeseen tilanteessa, jolloin mittausdata on hävinnyt.

6. Ruudunpäivityksen nopeuden mittaustuloksen automaattinen analysointi ohjelmistosta voisi osittain poistaa ongelmaa. Tämän voisi toteuttaa tunnistamalla ohjelmallisesti eri vikalähteiden profiileja kuvamateriaalista. Tämän lisäksi käyttäjälle voisi tarjota menetelmän tuloksen tarkistamiseen. Tämä voisi olla toteutettavissa visuaalisesti pyörittämällä analyysin kohteena oleva videoleike käyttäjälle hidastettuna. Hidastuksen lisäksi ohjelmisto voisi visualisoida kuvasta kohdat, joiden perusteella ruudunpäivitysnopeutta lasketaan.

7. Valikkojen tahattomaan piiloutumiseen voisi toimia sama korjaus kuin painikkeiden kohdallakin. Erottamalla niiden ulkoasu selkeästi ei-aktiivisista komponenteista käyttäjän olisi helpompaa havaita niiden olevan aktiivisia.

8. Tulosten toistettavuuden parantaminen ihmisen tehdessä ruudun vieritystä voi olla hyvin haasteellista ohjelmistossa. Asiaa voisi ehkä parantaa antamalla käyttäjälle menetelmä kahden eri mittauksen vertaamiseen. Tässä voisi toimia samantapainen visuaalinen esitys hidastetusta videosta kuin kohdassa numero 7.

9. Kuvan saamaa näyttöpinta-alaa voisi kasvattaa poistamalla niitä käyttöliittymäkomponentteja, jotka käyttäjä tuntee turhiksi omalta kohdaltaan. Yhtenä varteenotettavana ratkaisuna olisi käyttöliittymän muuttaminen sellaiseksi, missä käyttäjät itse voisivat määritellä haluamiensa komponenttien sijainnin ja koon käyttöliittymässä.

6.2.3 GOMS-mallin käyttö SPA:ssa

GOMS-mallia hyödyntämällä ylläoleva SPA:an käyttöesimerkki voidaan jakaa pienempiin kokonaisuuksiin. Käyttöesimerkin askeleet ovat seuraavat:

1. Käynnistetään SPA-ohjelmisto.
2. Käyttäjä asettaa mitattavan laitteen alustalle siten, että se on hyvässä asemassa kameraan nähden.
3. Käyttäjä tarkentaa ja kohdistaa kameran siten, että kohta, mistä vieritysnopeutta on tarkoitus mitata, on mahdollisemman terävästi näkyvissä SPA:n ruudulla.
4. Käyttäjä tekee haluamansa säädöt SPA-ohjelmistoon.
5. Käyttäjä määrittää mittauksen alaisen kohteen ruudulta SPA:n ruudulla olevasta videosyötteestä.
6. Käyttäjän ollessa valmis mittauksen aloitukseen käyttäjä painaa “Record”-painiketta.
7. Käyttäjä vierittää ruutua pyyhkäisemällä.
8. Mittauksen loppupisteessä käyttäjä painaa “Recording”-painiketta.
9. Käyttäjä odottaa analyysin päättymistä.
10. Käyttäjä tarkastelee SPA:n ilmoittamaa tulosta.
11. Halutessaan käyttäjä voi tarkentaa haluamaansa kohtaa aikajanalla ja käynnistää uuden analyysin haluamalleen kohdalle.

Välitavoitteet voidaan hahmottaa esimerkin tapauksesta seuraavasti:

1. Mitattavan laitteen valmistelu mittausta varten siten, että laite on käyttäjän syötettä odottavassa tilassa.

2. SPA-ohjelmiston valmistelu mittaustapahtumaa varten siten, että se odottaa käyttäjän "Record"-painikkeen painallusta.

3. Mittaus, joka aloitetaan painamalla "Record"-painiketta ja lopetetaan painamalla "Recording"-painiketta. Painikkeiden painamisen välillä mitattavan laitteen ruutua vieritetään.

4. Nauhoitetun videon analysointi. Tämä tapahtuu mittauksen jälkeen painamalla "Analyze"-painiketta.

SPA-ohjelmiston toiminnot ovat hyvin kohdistettuja tiettyyn käyttöön ja täten voidaan olettaa, että käyttäjälle sopivat tavoitteet ovat tiedossa suunnittelutyön alkuvaiheessa. Tämä luo GOMS-mallin käytölle edellytykset, ja siitä olisi hyvin todennäköisesti apua käyttöliittymän suunnittelussa. Käyttämällä CPM-GOMS-mallia sivutuotteena saataisiin myös käyttöohjeeseen proseduraaliset ohjeistukset SPA:n toimintojen hyödyntämiseen.

7 Käytettävyysongelmien korjaus ja välttäminen insinöörityökaluissa

On helppoa olettaa, että käytettävyysongelmien välttäminen olisi helpompaa, mikäli käytettävyyteen kiinnitettäisiin huomiota heti ohjelmiston alkumetreiltä asti. Tämä on kuitenkin useissa tapauksissa utopiaa nopeasti muuttuvassa tutkimus- ja tuotekehitysympäristössä, jossa pieniä työkaluja syntyy runsaasti. Jotkin näistä ovat hyvinkin tehtäväkohtaisia ja elinkaareltaan lyhyitä työkaluja, jolloin niiden käyttöliittymän käytettävyyteen ei ole järkevää tuhlata resursseja. Käytettävyysongelmia ja siitä seuraavaa tehtävän suorituksen tehokkuuden laskua saattaa syntyä, mikäli jokin näistä pikaisesti tehdyistä työkaluista jatkaakin elämäänsä pitemmän aikaa. Näissä tapauksissa käyttöliittymän korjaus saattaa olla viisas sijoitus jossain vaiheessa.

7.1 Lähtötilanteen määrittäminen

Insinöörityökalujen kanssa käytettävyysongelmien miettiminen hyvin todennäköisesti ei ole ensimmäisenä mielessä, mikäli kyseessä on työntekijän omaan käyttöön laatima työkalu. Tilanne ei välttämättä ole tämän suhteen erilainen edes pienen ryhmän kohdalla, koska ryhmän jäsenet voivat jakaa samantapaisen ajattelutavan asian suhteen. Mikäli työkalun tarve on määritetty yrityksessä korkeammalla tasolla, niin silloin on todennäköisempää, että käytettävyyttä joudutaan miettimään hieman. Tämä siksi, koska isomman yrityksen kohdalla yrityksen sisällä olevien ryhmien välillä saattaa olla hyvinkin suuria käsitteellisiä eroja käyttöliittymissä. On tärkeää kuitenkin muistaa, että käytettävyysongelmaa ei ole järkevää väkisin luoda muiden kuin kohdekäyttäjän tarpeista. Tämä korostuu varsinkin pienemmissä työkaluissa, joissa käyttäjäkunta on hyvin homogeeninen, jolloin käytettävyyden viilaaminen jokaiselle sopivaksi söisi valtavasti resursseja hyötyihinsä nähden.

Karkeasti jaettuna lähtötilanteita käytettävyysongelmien suhteen insinöörityökaluissa edellä mainituista syistä voisi olla määritettynä kolme seuraavaa:

1. Käyttöliittymä, joka on suunniteltu yhden henkilön tarpeisiin.

2. Käyttöliittymä, joka on suunniteltu toimimaan saman käyttettävyysnäkemyksen jakavien henkilöiden kanssa.

3. Käyttöliittymä, joka on alusta pitäen suunniteltu erilaisia taustoja omaavien henkilöiden kanssa.

Lähtötilanne vaikuttaa käytettävyyden parannusprosessissa käytettävien työkalujen ja menetelmien valintaan oleellisesti.

7.2 Käytössä olevan työkalun käytettävyyden parantaminen

Käytössä olevan työkalun kohdalla on se etu suunnittelutyötä ajatellen, että kyseisen työkalun käyttötapoja voidaan kartoittaa nykyisiltä käyttäjiltä. Työkalun käyttötavat ymmärtämällä suunnittelutyölle saadan vankka perusta, jonka pohjalta voidaan lähteä kehittämään työkalun käyttöliittymää näitä käyttötapoja paremmin tukevaksi. Vanhojen käyttäjien profiilien avulla voidaan myös ennustaa tulevien uusien käyttäjien vaatimuksia työkalun suhteen ja hyödyntää tätä tietoa suunnittelutyössä. Tämän lisäksi nykyistä toteutusta olisi hyvä käydä läpi esimerkiksi Nielsenin heuristiikkojen [Nielsen, 1993] kanssa, jolloin useimmat perustavanlaatuiset käytettävyysongelmat jäisivät kiinni ja joutuisivat muutoksen alle. Käytössä olleen työkalun kohdalla on myös se etu, että käyttöhistorian puitteissa työkalulla saavutettavien tavoitteiden pitäisi olla jollain asteella tiedossa. Tämä puolestaan mahdollistaa GOMS-mallin tehokkaan käytön. Mikäli kyseessä on yksinkertainen työkalu, jossa on yksinkertaiset askeleet tehtävän suorituksessa, voitaisiin yksinkertaisimmillaan käyttää hyväksi KLM-mallia avuksi uusien käyttöä tehostavien muutoksien kokeiluun ennen varsinaista toteutusta. Laajemmissa työkaluissa voisi puolestaan olla hyväksi käyttää esimerkiksi CMN-GOMS-mallia, koska sen prosessinkuvaustekniikoiden avulla voitaisiin ennustaa käyttäjän tulevat toimet sekä nähdä tehokkaimmat suorituspolut. Käytössä olleen työkalun käytettävyyttä parannettaessa on hyvä edetä kuitenkin varovasti, ettei mitään toimivaa menetelmää romuteta muutoksia tehdessä, vaikka se saattaisi sotia käytettävyydenarvioinnin heuristiikkoja vastaan.

7.3 Uuden työkalun suunnittelun ensiaskeleet

Täysin uutta työkalua luotaessa on hyvä tiedostaa tuleva käyttäjäkunta ja mitoittaa käyttöliittymän suunnittelu sellaiseksi, että se tuottaa hyvän käytettävyyden potentiaalisille käyttäjille. Erilaisten suunnitelmavaihtoehtojen vertailu mallien avulla on syytä tehdä harkiten. Esimerkkinä voidaan mainita aikaisemmin mainittu GOMS-malli, jonka käyttö on kyseenalaista, mikäli työkalulle asetettuja tavoitteita ei kyetä määrittämään suunnittelutyön alkuvaiheessa tarpeeksi tarkasti. Omaan käyttöön tulevaa työkalua suunniteltaessa on suotavaa tehdä riskianalyysi siitä, onko työkalulla potentiaalia päätyä muiden käytettäväksi myöhemmin. Tällainen riskin kohdalla tällöin kannattaa käytettävyyteen muidenkin silmissä kiinnittää huomiota, koska käytettävyyden huomioiminen on helpompaa alkuvaiheessa. Tämä ei poissulje sitä, että työkalun tekijä ei voisi kehittää käyttöliittymää omien mieltymyksien mukaan omia työtapoja tukevaksi. Riskin ollessa minimaalinen tai olematon tekijä voi keskittyä järkevän ajankäytön nimissä tekemään käyttöliittymästä täysin omia toimintatapoja tukevan. Tässäkin tapauksessa yleisen käytettävyyden huomioiminen toteutuksessa olisi toivottavaa, mutta ei välttämätöntä, mikäli aikarajat eivät anna myöden.

7.4 Käytettävyysongelmien välttäminen

Käytettävyysongelmien välttäminen muidenkin ongelmien tapaan on helpointa mahdollisimman varhaisessa vaiheessa, ennen kuin niistä kasvaa suurempia ongelmia. Insinöörityökalujen käytettävyyttä suunniteltaessa olisi hyvä noudattaa Nielsenin kymmentä kuuluisaa heuristiikkaa [Nielsen ja Mack, 1994], joita voisi pitää myös hyvän käytettävyyden peukalosääntöinä. Insinöörityökaluihin mukautettuina ne voisivat kuulua seuraavasti:

1. Järjestelmän tilan näkyminen. Insinöörityökalun tulee aina näyttää tilansa selkeästi käyttäjälle. Tämän asian ollessa kunnossa, käyttäjän ei tarvitse nähdä erityistä vaivaa selvittääkseen työkalun nykytilan.

2. Järjestelmän ja ympäröivän maailman yhtäläisyys. Käyttäjän silmissä järjestelmän tulisi olla sopusoinnussa ympäröivän maailman kanssa. Tämän asian ollessa kunnossa käyttäjän ei tarvitse tehdä kontekstin vaihtoa käyttäessään järjestelmää ja sen lähellä olevia muita järjestelmiä tai asioita.

3. Käyttäjän kontrolli ja vapaus. Järjestelmän ei tulisi pakottaa käyttäjää toimimaan tietyn kaavan mukaan, ellei siihen ole perusteltua syytä tavoit-

teen onnistumisen vuoksi. Käyttäjän vapautta tukemalla käyttöliittymä voi mahdollistaa käyttäjän kokeilemaan erilaisia työskentelytapoja, jotka voivat olla hänen kannaltaan parempia valmiiksi suunniteltujen tapojen sijaan.

4. Johdonmukaisuuden ja standardien kunnioitus. Huomioimalla johdonmukaisuuden periaatteen suunnittelija mahdollistaa sen, että järjestelmä toimii samojen periaatteiden mukaisesti joka puolella. Tämän ollessa kunnossa käyttäjän ei tarvitse opiskella erilaisia toimintatapoja saman järjestelmän sisällä. Standardit huomioimalla saavutetaan aikaisemman kohdan mainitsemaa yhdennäköisyyttä ympäröivän maailman kanssa.

5. Virhetilanteeseen joutumisen välttäminen. Mikäli suunnitteluvaiheessa on nähtävissä, että jonkin toimintosarjan yhtenä toteutumana on potentiaalinen virhetilanne käyttäjän näkökulmasta, on siitä ilmoitettava käyttäjälle ennen virheen tapahtumista ja tarjottava vaihtoehtoista reittiä ongelman kiertämiseen, mikäli mahdollista.

6. Tunnistettavuus muistamisen sijaan. Tätä kunniottamalla käyttöliittymän suunnittelija minimoi käyttäjän muistikuormaa tekemällä käyttöliittymän komponenteista mahdollisimman näkyviä toiminallisuutensa puolesta. Käyttäjän ei näin tarvitsisi palata käyttöohjeen pariin käytön aikana.

7. Estetiikka ja minimalistisuus. Käyttöliittymässä ei tulisi olla mitään ylimääräistä, koska se vie huomiota tehtävän kannalta relevantilta komponentilta.

8. Auta käyttäjää tunnistamaan ja analysoimaan virhetilanne. Mikäli virhetilanteeseen päädytään, järjestelmän pitäisi kyetä antamaan käyttäjälle selväkielinen ohjeistus virheen syystä ja mahdollisesti kertoa, miksi kyseiseen tilaan päädyttiin. Tämän lisäksi järjestelmän olisi hyvä kyetä antamaan käyttäjälle myös tieto siitä, että miten virhetilanteesta selvitään.

9. Dokumentaatio ja avustustoiminto. Tarjottavan dokumentaation tulisi olla selkeää ja konkreettisia esimerkkejä sisältävää. Lisäksi halutun tiedon haun dokumentaatiosta tulisi olla vaivatonta. Ajonaikaisen avuntarjoamisen tulisi puolestaan tarjota kyseiseen tilanteeseen relevanttia tietoa ja olla helposti saatavilla kyseisessä tilanteessa. Muiden kahdeksan kohdan ollessa kunnossa voitaisiin olla kuitenkin lähellä ideaalitilannetta, jossa käyttäjä kykenisi käyttämään järjestelmää tehokkaasti ilman dokumentaatiotakin.

Pitämällä nämä peukalosäännöt mielessään työkalun käyttöliittymän suunnittelija välttäisi pahimmat käytettävyysongelmat. Hyvää käytettävyyttä on helppo perustella paremman työkalun käytettävyyden lisäksi myös sen luomalla ajansäästöllä, kun käyttäjien työskentely on tehokkaampaa kuin huonomman käytettävyyden omaavalla työkalulla. Pienikin käytettävyysongelman aiheuttama ajanhukka saattaa muodostua merkittäväksi, mikäli sama käyttävyysongelma kohdataan liian usein. Suurinta ajallista hyötyä käyttötilanteessa edellä mainituista olisi mahdollista saada esimerkiksi "Tunnistettavuus muistamisen sijaan"-heuristiikan avulla, koska laajassa ohjelmistossa käyttäjälle asetetut muistamisen vaatimukset voivat olla muuten hyvin suuret ja tätä heuristiikkaa noudattamalla sen voi välttää. Tässä tapauksessa ajansäästö syntyy käyttäjän puolelta, kun aikaa ei kulu ylimääräiseen asian muistelemiseen. Toisena vastaavana esimerkkinä voidaan pitää "Auta käyttäjää tunnistamaan ja analysoimaan virhetilanne"-heuristiikkaa, jota noudattamalla järjestelmän päätyessä virhetilaan käyttäjälle ei jäisi epätietoisuutta tilanteen luonteesta. Tämän avulla myös ajansäästöä syntyy, kun käyttäjä osaa välttää virhetilaan johtavat syyt tulevilla käyttökerroilla.

8 Yhteenveto

Käytettävyyden huomioiminen järjestelmää suunniteltaessa on hyvä aloittaa käytettävyyteen liitettyjen heuristiikkojen avulla. Niitä hyödyntämällä voidaan välttää pahimmat yleiset käytettävyysongelmat. On tärkeää kuitenkin huomioda, että käyttöliittymän tehokkuutta tavoiteltaessa insinöörityökaluissa ajaudutaan ristiriitaan heuristiikkojen kanssa jossain tapauksissa. Esimerkkinä tällaisesta ristiriidasta voidaan pitää käyttöliittymän yksinkertaistamista mahdollisimman pitkälle Nielsenin heuristiikkoja noudattamalla. Yksinkertaistamisen seurauksena monia eri toimintoja sisältävä insinöörityökalun käytettävyyden tehokkuus saattaisi laskea kokeneella käyttäjällä, koska yhden sekavan pääkäyttöliittymän sijaan käyttäjän olisi siirryttävä eri näkymien tai valikkojen kautta haluamiensa toimintojen pariin. Tämä ei kuitenkaan tarkoita sitä, että molempia ei voitaisi saavuttaa luovilla ratkaisuilla. Tutkielman luvussa 3.1 esitelty Googlen hakukone toimii yhtenä tälläisenä hyvin pitkälle yksinkertaistettuna käyttöliittymänä, jossa kehittyneemmät toiminnot ovat saatavilla hakukenttään syötettävillä parametreillä ilman ylimääräisen kompleksisuuden lisäämistä varsinaiseen käyttöliittymään.

GOMS-malli tarjoaa yhden varteenotettavan työkalun tehokkuuden ja yleisen käytettävyyden tasapainon löytämiseen insinöörityökaluissa. Mallin käytön edellytykset insinöörityökalujen kohdalla täyttyvät, koska kohdekäyttäjät ja halutut tavoitteet ovat yleensä jollain tasolla tiedossa ennen työkalun suunnittelun tai toteuttamisen aloittamista. Tämän lisäksi insinöörityökalun kohdekäyttäjillä voi-daan yleensä olettaa olevan kohtuullinen tietous tavoitteen saavuttamiseen tarvituista askelista, jolloin GOMS-mallin heikkous aloittelijan kognitiivisten prosessien hahmottamisessa ei esiinny niin voimakkaasti. GOMS-mallin luominen vaatii resursseja huolellisesti käytettynä, mutta sen käyttö saattaa säästää resursseja ajallisesti. Tämä korostuu varsinkin työkaluissa, joiden käyttöaste on korkea. GOMS-mallin käyttöön sijoitettujen resurssien takaisinmaksu on tällöin hyvinkin nopeaa ja mallin käyttö on hyvin perusteltua. GOMS-mallin käyttö tehostaa myös itse työkalun kehitystyötä, koska se mahdollistaa erilaisten käyttöliittymäratkaisujen kokeilemisen ennen toteuttamista.

Olemassa olevan insinöörityökalun kanssa käytettävyyden parantaminen onnistuu soveltamalla samoja keinoja kuin uudenkin työkalun kanssa. Näiden keinojen lisäksi käytössä olleen insinöörityökalun kanssa on mahdollista kerätä haastatteluiden ja muiden käyttöanalyysien kautta saatavaa tietoa muutostarpeista toimin-

nan tehostamiseen. Nämä tiedot parantavat GOMS-mallin käyttöä ja sen avulla tuotettujen ratkaisujen laatua kohdekäyttäjäkunnan silmissä. Käytössä olleen työkalun muutoksia suunniteltaessa on huomioitava mahdollinen muutosvastarinta, joka saattaa mitätöidä oletetut hyödyt muutoksista. Tätä ongelmaa voidaan ehkäistä myös haastatteluista ja käyttöanalyyseistä saatavilla tiedoilla.

Tutkimuksessa esiteltiin keinoja käytettävyyden parantamiseen ja niiden vaikutuksia työkalun tehokkuuteen. Yhtenä tällaisena keinona esiteltiin GOMS-malliperheeseen kuuluvan KLM-mallin toimivuutta käytössä olevan OptoFidelityn insinöörityökalun käyttöliittymän kehittämisessä kappaleessa 6.1.4. Ehdotettujen käyttöliittymämuutosten vaikutusta KLM-mallin avulla voitiin arvioida ilman varsinaista toteutusta. Mallin avulla tehtyjen laskelmien mukaan ehdotetuilla muutoksilla olisi mahdollista parantaa tehokkuutta käytettävyyden kautta. Huomionarvoista esitellyssä tapauksessa on, että yksittäistä käyttökertaa tarkasteltaessa aikaa ei säästy suurta määrää. Muutoksiin tarvittavien resurssien käyttö on kuitenkin perusteltavissa, koska muutoksien kohteena olevaa toiminallisuutta käytetään paljon ja pieni ajansäästö yhdellä kierroksella kertautuu tehtyjen kierrosten mukaan.

Viiteluettelo

[1] [Arad, 2012] Ayala Arad, *Past Decisions Do Affect Future Choices: An Experimental Demonstration*, Organizational Behavior and Human Decision Processes Vol 121, No 2, 267 - 277, 2012.

[2] [Card et al., 1980a] Stuart Card, Thomas P. Moran, Allen Newell, *The Keystroke-Level Model for user performance time with interactive systems*, Communications of the ACM Vol 23, No 7, 396 - 410, 1980a.

[3] [Card et al., 1980b] Stuart Card, Thomas P. Moran, Allen Newell, *Computer text-editing: An informationprocessing analysis of a routine cognitive skill.*, Cognitive Psychology 12, 32 - 74, 1980b.

[4] [Card et al., 1983] Stuart Card, Thomas P. Moran, Allen Newell, *The Psychology of Human-Computer-Interaction.*, Lawrence Erlbaum, Hillside, N.J, ISBN 0898592437, 1983.

[5] [Carroll, 2003] John M. Carroll, *HCI Models, Theories and Frameworks. Toward A Multidisciplinary Science*, Morgan Kaufmann Publishers, ISBN 1-55860-808-7, 2003.

[6] [Frokjaer et al., 2000] Erik Frokjaer, Morten Hertzum, Kasper Hornbaek, *Measuring usability: are effectiveness, effiency, and statisfaction really correlated*, Proceedings of the CHI 2000, 345 - 353, 2000.

[7] [Gardete, 2014] Pedro M. Gardete, *Understanding Social Effects in the In-Flight Marketplace: Characterization and Managerial Implications*, Journal of Marketing Research Vol 52, No 3, 360 - 374, 2014.

[8] [Gong, Elkerton, 1990] Richard Gong, Jay Elkerton, *Designing Minimal Documentation Using a GOMS Model: A Usability Evaluation of an Engineering Approach*, ACM CHI 1990 Proceedings, 99 - 107, April 1990.

[9] [ISO 9421] ISO 9421, *Näyttöpäätteillä tehtävän toimistotyön ergonomiset vaatimukset. Osa 11: Käytettävyyden määrittely ja arviointi*, Suomen Standardoimisliitto, Helsinki, 1998.

[10] [ISO 9126] ISO 9126, *Software product Evaluation: Quality Characteristics and Guidelines for their Use*, Suomen standardoimisliitto, 1991.

[11] [ISO 13407] ISO 13407, *Vuorovaikutteisten järjestelmien käyttäjäkeskeinen suunnitteluprosessi*, Suomen standardoimisliitto, Helsinki, 1999.

[12] [Kieras et al., 1994] Bonnie E. John, David E. Kieras, *The GOMS Family of Analysis Techiniques: Tools for Design and Evaluation*, School of COmputer Science Camegie Mellon University Pittsburgh, PA 15213, CMU-HCII, 94 - 106, 1994.

[13] [Kieras et al., 1996a] Bonnie E. John, David E. Kieras, *Using GOMS for User Interface Design and evaluation: Which Technique?*, ACM Transactions on Computer-Human Interaction, Vol 3, No 4, 287 - 319, 1996a.

[14] [Kieras et al., 1996b] Bonnie E. John, David E. Kieras, *Using GOMS for User Interface Design and evaluation: Comparison and Contrast.* ACM Transactions on Computer-Human Interaction Vol 3, No 4, 320 - 351, 1996b.

[15] [Kieras, 2005] David E. Kieras, *GOMS Models - Simplified Cognitive Architectures*, The University of Michigan, http://hccedl.cc.gatech.edu/documents/82_Kieras_2a_GOMS_Architectures.pdf (luettu 10.02.2015), 2005.

[16] [Kieras et al., 1985] David E. Kieras, Peter G. Polson, *An approach to formal analysis of user complexity*, International Journal of Man-Machine Studies. Vol. 22, 365 - 394. 1985.

[17] [Kieras, 1988] David E. Kieras, *Towards a practical GOMS model methodology for user interface design.* Teoksessa *The Handbook of human-computer interaction, 135 - 158*, M. Helander (toim.). Amsterdam, North-Holland, 1988.

[18] [Nielsen, 1993] Jakob Nielsen, *Usability Engineering*, Academic Press Inc, ISBN: 0-12-518406-9, 1993.

[19] [Nielsen, 1994] Jakob Nielsen, Robert L. Mack, *Usability Inspection Methods*, John Wiley & Sons, New York, ISBN 0-471-01877-5, 1994.

[20] [Norman, 1988] Donald A. Norman, *The Design of Everyday Things*, The MIT Press, ISBN: 1452654123, 1988.

[21] [Optofidelity, 2014] OptoFidelity Oy, *OptoFidelity WatchDog - User Manual 2014 Q4*, OptoFidelity, 2014.

[22] [Raita ja Oulasvirta, 2011] Eeva Raita, Antti Oulasvirta, *Too Good to be Bad: The Effect of Favorable Expectations on Usability Perceptions*, Cognitive Ergonomics for Situated Human-Automation Collaboration, Vol 23, No 4, 363 - 371, 2011.

[23] [Saponas et al., 2006] Scott T. Saponas, Madhu K. Prabaker, Gregory D. Abowd, James A. Landay, *The Impact of Pre-Patterns on the Design of Digital Home Applications*, DIS 2006 Proceedings of the 6th conference on Designing Interactive systems, 189 - 198, ISBN: 1-59593-367-0, 2006.

[24] [Seffad ja Metzker, 2004] Ahmed Seffad, Eduard Metzker, *The Obstacles and Myths of Usability and Software Engineering*, Communications of the ACM, Vol 47, No 12, 2004.

[25] [Sharp et al., 2007] Helen Sharp, Yvonne Rogers, Jenny Preece, *Interaction Design: Beyond Human-Computer Interaction, 2nd Edition*, John Wiley & Sons Ltd, 2007.

[26] [Tourangeau et al., 1997] Roger Tourangeau, Tom W. Smith, Kenneth A. Rasinski, *Motivation to Report Sensitive Behaviors on Surveys: Evidence From a Bogus Pipeline Experiment*, Journal of Applied Social Psychology, Vol 27, No 3, 209 – 222, 1997.

[27] [The University of Michigan, 1988] The University of Michigan, *User's manual for the two dimensional static strength prediction program*, Ann Arbor MI, 1988.